

SCHEME : K

Name : _____

Roll No. : _____ Year : 20__ 20__

Exam Seat No. : _____

LABORATORY MANUAL FOR LINUX BASICS (312001)



COMPUTER ENGINEERING GROUP



**MAHARASHTRA STATE BOARD OF
TECHNICAL EDUCATION, MUMBAI**
(Autonomous) (ISO 9001: 2015) (ISO/IEC 27001:2013)

A Laboratory Manual

for

Linux Basics

(312001)

Second Semester

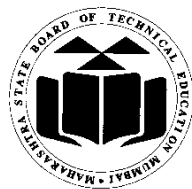
Diploma in Computer Engineering/
Computer Science & Engineering/ Data
Sciences/ Computer Hardware &
Maintenance/Information Technology/
Computer Science & Information
Technology



Maharashtra State

Board of Technical Education, Mumbai

(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)



Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)
4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai -400051.



Maharashtra State Board of Technical Education Certificate

This is to certify that Mr. / Ms.
Roll No.....of Second Semester of Diploma in
..... of Institute
.....
(Code) has attained predefined laboratory learning
outcomes (LLOs) satisfactorily in course LINUX BASICS (312001)
for the academic year 20.....to 20..... as prescribed in the
curriculum.

Place:
Date:

Enrollment No.:
Exam Seat No.:

Course Teacher

Head of the Department

Principal



Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative ‘K’ Scheme curricula for engineering Diploma programmes with outcome-based education as the focus and accordingly, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher, instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a ‘*vehicle*’ to develop this industry identified competency in every student. The practical skills are difficult to develop through ‘chalk and duster’ activity in the classroom situation. Accordingly, the ‘I’ scheme laboratory manual development team designed the practical’s to *focus on outcomes*, rather than the traditional age old practice of conducting practical’s to ‘verify the theory’ (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read the concerned practical procedure that they will do the next day and understand minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. Students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

Operating systems are an essential part of any computer system. Similarly, a course on operating systems is an essential part of any computer group. We hope that students will also find it useful. It provides a clear description of practical performance, execution and working of Operating System.

The lab manual development team wishes to thank MSBTE who took initiative in the development of curriculum re-design project and implementation and also acknowledge the

contribution of individual course experts who have been involved in laboratory manual as well as curriculum development (I scheme) directly or indirectly.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome.

Lab Manual Development Team

Programme Outcomes (POs) to be achieved through Practicals of this Course

Following programme outcomes are expected to be achieved significantly out of the ten programme outcomes and Computer Engineering and Information Technology programme specific outcomes through the practicals of the course on **Linux Basics**.

PO 1. Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO 2. Problem analysis: Identify and analyse well-defined engineering problems using codified standard methods.

PO 3. Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO 4. Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO 5. Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO 6. Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

PO 7. Life-long learning: Ability to analyse individual needs and engage in updating in the context of technological changes.

Practical - Course Outcome Matrix

S. No.	Title of the Practical	CO a.	CO b.	CO c.	CO d.	CO e.
1	Install the Linux operating system.	√				
2	Execute general-purpose Linux commands.		√			
3	Execute the following general-purpose Linux commands.		√			
4	Execute the following file and Directory manipulation commands			√		
5	Execute the following file and Directory manipulation commands			√		
6	Execute Linux commands for compressing decompressing and archiving files.			√		
7	change file and directory permissions.			√		
8	Use vi editor to create and edit files				√	
9	Use wildcard characters				√	
10	Use of Pipes in Linux.				√	
11	Create input and output redirection in Linux.				√	
12	Execute the filters commands in Linux.					√
13	Execute filters commands in Linux.					√
14	Execute the Shell scripts.					√
15	Execute the Shell script by using the if statement.					√
16	Execute a Shell script by using the while loop					√
17	Execute a Shell script by using the for loop					√

List of Industry Relevant Skills

The following industry relevant skills of the competency “Manage operations of Operating System” are expected to be developed in you by performing practicals of this laboratory manual.

1. To understand the basics of Linux operating system fundamentals and its open-source nature.
2. Basic Scripting Skills for automating tasks and creating custom shell scripts.
3. Ability to perform file operations and manipulate directories.

Brief Guidelines to Teachers

Hints regarding strategies to be used

1. Teacher shall explain prior concepts to the students before starting each experiment.
2. For practical's requiring tools to be used, teacher should provide the demonstration of the practical emphasizing the skills, which the student should achieve.
3. Involve students in the activities during the conduct of each experiment.
4. Teachers should give opportunity to students for hands-on after the demonstration.
5. Assess the skill achievement of the students and COs of each unit.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
8. Teacher may provide additional knowledge and skills to the students even though that may not be covered in the manual but are expected from the students by the industries.
9. Teacher may suggest the students to refer additional related literature of the reference books/websites/seminar proceedings etc.
10. During assessment teacher is expected to ask questions to the students to tap their knowledge and skill related to that practical.

Instructions for Students

Student shall read the points given below for understanding the theoretical concepts and practical applications.

1. Students shall listen carefully the lecture given by teacher about importance of subject, learning structure, course outcomes.
2. Students shall organize the work in the group of two or three members and make a record of all observations.
3. Students shall understand the purpose of experiment and its practical implementation.
4. Students shall write the answers of the questions during practical.
5. Student should feel free to discuss any difficulty faced during the conduct of practical.
6. Students shall develop knowledge of Operating System fundamental and manipulation skills as expected by the industries.
7. Student shall attempt to develop related hands on skills and gain confidence.
8. Students shall refer technical magazines; websites related to the scope of the subjects and update their knowledge and skills.
9. Students shall develop self-learning techniques.
10. Students should develop habit to submit the write-ups on the scheduled dates and time.

Content Page
List of Practicals and Progressive Assessment Sheet

S. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks(25)	Dated sign.of teacher	Remarks (if any)
1.	Install the Linux operating system.	01					
2.	Execute general-purpose Linux commands.	05					
3.	Execute the following general-purpose Linux commands.	12					
4.	Execute the following file and Directory manipulation commands	18					
5.	Execute the following file and Directory manipulation commands	25					
6.	Execute Linux commands for compressing decompressing and archiving files.	31					
7.	change file and directory permissions.	37					
8.	Use vi editor to create and edit files	42					
9.	Use wildcard characters	52					
10.	Use of Pipes in Linux.	57					
11.	Create input and output redirection in Linux.	62					
12.	Execute the filters commands in Linux.	67					

S. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks(25)	Dated sign.of teacher	Remarks (if any)
13.	Execute filters commands in Linux.	75					
14	Execute the Shell scripts.	80					
15	Execute the Shell script by using the if statement.	83					
16	Execute a Shell script by using the while loop	89					
17	Execute a Shell script by using the for loop	93					
Total Marks							

- To be transferred to Proforma of CIAAN-2023.

Experiment No 01: Install and configure the Linux operating system.

- I. Practical Significance:** Linux is the base of many of open source operating systems designed to replace Windows and Mac OS. It is free to download and install on any computer. Because it is open source, there are a variety of different versions, or distributions, available developed by different groups. To prevent hacking attempts, many organizations keep their Linux operating systems private. Many others make their variations of Linux available publicly so the whole world can benefit at large.
- II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:**
To understand the basics of Linux operating system fundamentals and its open-source nature.
- III. COURSE LEVEL LEARNING OUTCOMES (COS):**
CO1 - Install Linux operating system.
- IV. LABORATORY LEARNING OUTCOME:**
Install and configure the Linux operating system.
- V. Relevant Affective Domain related outcome(s)**
1. Follow precautionary measures.
 2. Follow installation steps.
 3. Follow ethical practices.
- VI. Relevant Theoretical Background**
- Step:1 Download the ISO file.
- Step:2 Boot your system with Bootable DVD / USB drive. To start the installation click on "Install Ubuntu"
- Step:3 Check Install Prerequisite
- Step:4 Select the Installation Type
- Step: 5 Select your respective Time Zone
- Step:6 Select your respective Keyboard Layout
- Step:7 Set the Hostname of your system and User credentials that will be used after installation. Installation has started. Once the installation is completed, it will ask to restart the Machine. Click on "Restart Now"
- Step:8 Login Screen after reboot. Use the same user and its credentials that we have set during the installation. We will get below screen after entering the credentials. Ubuntu Installation is Completed Now. Similarly any open source installation shall be considered.

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure

Teacher must assign a separate program statement to students. Install and configure Linux (or similar) operating system on your computer. Write down the steps for same.

IX. Result(s)

.....

.....

.....

.....

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. What are different versions of Linux operating system?
2. Enlist the steps for booting the operating system.
3. State names of latest multiuser operating system and its advantages.

(Space for answer)

[illegible]

This image shows a full page of a notebook or worksheet. It features approximately 30 horizontal dotted lines spaced evenly across the page, providing a guide for handwriting practice. The lines are light gray and extend from the left margin to the right edge of the page. There are no margins, text, or other markings present.

XI. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>

2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 02: Execute general purpose Linux commands.

I. Practical Significance: General purpose commands are inbuilt programs that can be invoked in multiple ways. These commands work interactively from a terminal. A terminal that provides a command line interface using a shell program.

II. Industry / Employer Expected Outcome:

- 1.To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2.Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3.Ability to perform file operations and manipulate directories.

III. Course Level Learning Outcomes (COs):

CO2 - Execute general purpose commands of the Linux operating system.

IV. Laboratory Learning Outcome:

Execute general purpose commands cal, date, echo, printf, bc, script, mailx, man, clear

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI. Relevant Theoretical Background

1. **cal** : command is used to display a calendar, or some part of it. If no options are specified then it displays current month.

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

\$cal -1 :show only a single month (default)

\$cal -3 :show three months spanning the date

\$cal -s :Sunday as first day of week

\$cal -m :Monday as first day of week

\$cal -j :output Julian dates

\$cal -y :show the whole year

\$cal -w :show US or ISO-8601 week numbers

2. **date**: date command is used to display the system date and time. date command is also

used to set date and time of the system. You must be the super-user (root) to change the date and time.

Syntax: date [OPTION]... [+FORMAT]

3. **echo:** echo command in Linux is a built-in command that allows users to display lines of text or strings that are passed as arguments. It is commonly used in shell scripts and batch files to output status text to the screen or a file.

Syntax : echo [option] [string]

echo " POLYTECHNIC COLLEGE " output : POLYTECHNIC COLLEGE

4. **printf:** printf command in Linux is used to display the given string, number or any other format specifier on the terminal window. It works the same way as “printf” works in programming languages like C.

Syntax:

\$printf [-v var] format [arguments]

printf can have format specifiers, escape sequences or ordinary characters.

Format Specifiers: The most commonly used printf specifiers are %s, %b, %d, %x and %f.

Examples: %s specifier: It is basically a string specifier for string output.

\$printf "%s\n" "Hello, World!"

%d specifier: It is an integer specifier for showing the integral values. \$ printf "%d\n" "213"

5. **bc :** command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

You can use bc command in bash or shell script also for evaluating arithmetic expressions.

Input : \$ echo "12+5" | bc Output: 17

Input : \$ echo "10^2" | bc Output : 100

6. **script :** command in Linux is used to make typescript or record all the terminal activities.

After executing the script command it starts recording everything printed on the screen including the inputs and outputs until exit.

By default, all the terminal information is saved in the file typescript , if no argument is

given.

Syntax: script [options] [file]

If script command is used with filename then all the recorded terminal information is saved in the given file location.

7. **mailx:** Linux has an in built Mail User Agent program called mailx. As the name suggests, it is a console application that is used for sending and receiving emails. The mailx utility is an enhanced version of the mail command. Along with the functionality provided by the original mail command, it provides extra features like the ability to send attachments by using the -a flag. The mailx command is available from a variety of different packages: `bsd-mailx`, `heirloom-mailx`, `mailutils`

Installing **mailx** For Ubuntu/Debian: **`sudo apt-get install bsd-mailx`**

Sending an Email

Writing the message directly in the command line:

To send a simple email, use the “-s” flag to set the subject in quotes which is followed by the email of the receiver. After this, mailx waits for the content of the email. To enter new lines, keep hitting enter. After the content is written, press Ctrl+D.

```
$ mail -s "A mail sent using mailx" msbte.blp@bte.ac.in
```

Hey student ,

Hope you're fine these days

Thanks

8. **man:** The “man” command, short for manual, is a powerful tool in the Linux operating system that allows users to access detailed information about various commands, utilities, and system calls.

Syntax : `man [option] [command]`

For example, to view the manual for the “ls” command, you would enter: **`man ls`**

9. **clear :**clear is a standard Unix computer operating system command that is used to clear the terminal screen.

Syntax: `clear`

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

1. Write down different options of cal command. (Use \$man cal)

2. Write options of date command. (Use \$man date)

IX. Result(s)

.....
.....
.....
.

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. How you record all the following activities performed by the user.
2. Give a command to display calendar for month of February.
3. Give single statement command to display the calendar of previous, current and next month.

(Space for answer)

[illegible]

XI. Exercise:

1. What is output of following commands?

a. \$cal 04 2023

.....

.....

b. \$date "+Today's information: %D and %B"; cal

.....

.....

.....

c. `$date "+My clock is showing %H hours, %M minutes and %S seconds"`

.....

d. \$cal -3

.....
.....
.....

e. \$x=echo "var=500;var%=7;var" | bc

\$ echo \$x

.....

f.

g. \$ echo "var=10;var++" | bc

.....

g. printf "%f\n" "20.20"

.....

.....

(Attach page for output)

2. Give the syntax of commands for displaying the output. (use date)

i. This is Month of the year.....(Abbreviation for month and
4-digit year)

ii. This is the 'th day of this year.

(Space for answer)

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XII. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 03: Execute general-purpose Linux commands

- I. Practical Significance:** General purpose commands are inbuilt programs that can be invoked in multiple ways. These commands work interactively from a terminal. A terminal that provides a command line interface using a shell program.
- II. Industry / Employer Expected Outcome:**
- 1.To understand the basics of Linux operating system fundamentals and its open-source nature.
 - 2.Basic Scripting Skills for automating tasks and creating custom shell scripts.
 - 3.Ability to perform file operations and manipulate directories.
- III. Course Level Learning Outcomes (COs):**
CO2 - Execute general purpose commands of the Linux operating system.
- IV. Laboratory Learning Outcome:**
Execute general purpose commands 1)passwd 2)who 3)whoami 4)uname 5)tty 6)stty 7)ps 8)kill 9) sleep
- V. Relevant Affective Domain related outcome(s)**
1. Follow precautionary measures.
 2. Follow naming conventions
 3. Follow ethical practices.
- VI. Relevant Theoretical Background**
1. **passwd:** passwd command in Linux is a powerful tool that allows system administrators and users to manage password-related tasks. Its primary purpose is to change user passwords, but it offers additional functionalities such as updating password aging policies, unlocking accounts etc.

syntax: passwd [options] [username]
[options] = include various parameters to customize the password-changing process.
[username] = the target user account for which you want to change the password
 2. **who :** It is used to display who are the users connected to our computer currently
Syntax :who [options] [filename]

Option	Description
-a	Same as -b -d -login -p -r -t -T -u
-b	Time of last system boot
-d	Print dead processes
-H	Print line of column headings
-l	Print system login processes
-m	Only hostname and user associated with stdin
-p	Print active processes spawned by init
-q	All login names and number of users logged on

3. **whoami** : Display the details of the current working directory

Syntax: whoami [OPTION]

1.--help Option :It gives the help message and exit.

2.--version: Option: It gives the version information and exit.

3. -u option display UID (User ID) This option displays the UID (User ID) instead of the username.

4. -e This option shows the effective user ID, providing information about the user's privileges.

4. **uname**: displays the information about the system.

```
goelashwin3d@Ash:~$ uname -a
Linux Ash 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018 x86_64 x
86_64 x86_64 GNU/Linux
goelashwin3d@Ash:~$
```

uname [OPTIONS]

1. -a : Displays all available information
2. -s : Shows the kernel name
3. -r : It prints the kernel release date.

5. **tty**: command is to display the name of the current terminal.

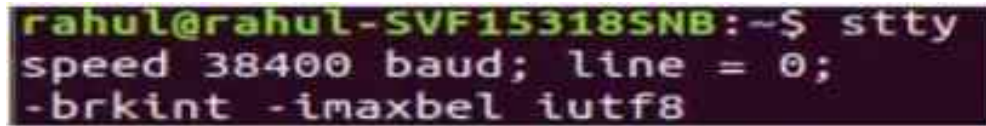
Syntax: tty [OPTION]

- 1) -a:which displays the name of all currently active terminals,
- 2) -h: which displays the full path name of the terminal device file.
- 3) --help: It will display the help message and exit.

4)--version: Prints the version information and exits.

6. **stty**: **stty** command in Linux is used to change and print terminal line settings. Basically, this command shows or changes terminal characteristics.

Syntax: stty



```
rahu1@rahu1-SVF153185NB:~$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel iutf8
```

7. **ps** : Display the characteristics of a process. (ie. terminal number,time required, PID no, and command name)

\$ps -f Full listing showing PPID of each process.

\$ps -u username Displays processes of user 'username'

\$ps -a Processes of all users

\$ps -e Processes including user and system processes.

8. **kill** : Used to stop execution of particular process by sending an interrupt signal to the process.

Syntax : kill [signal]PID

\$kill 0 Kills all the processes on the terminal except the login shell by special argument '0'

\$kill 120 230 234 Kills three processes with pid 120 230 234

\$kill -9 0 Kills all processes including login shell

\$kill -9 \$\$ Kills login shell

- 9.**sleep** : This command pauses the execution for an amount of time which is defined by *NUMBER*.

Syntax: sleep NUMBER[SUFFIX]

NUMBER represents the time duration for which the command should sleep.

SUFFIX can be used to specify the unit of time (s for seconds, m for minutes, h for hours, etc.)

Ex. Sleep 30

VII. Program code

3. Write down different options of kill command.

4. Write options of uname command.

VIII. Result(s)

IX. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

4. Name the command which terminate the process
5. Write a command for full listing of all running process with output
6. Write use of passwd command.
7. To delay process for specific time command used is _____ (ps/sleep).

(Space for answer)

X. Exercise:

3. What is output of following commands?

a. \$ps -f

.....
.....
.....

b. \$stty

.....
.....
.....

c. \$tty -s

.....
.....

d. \$whoami

.....
.....

e.\$uname -s

.....

h. \$sleep 100

.....

(Attach page for output)

(Space for answer)

XI. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

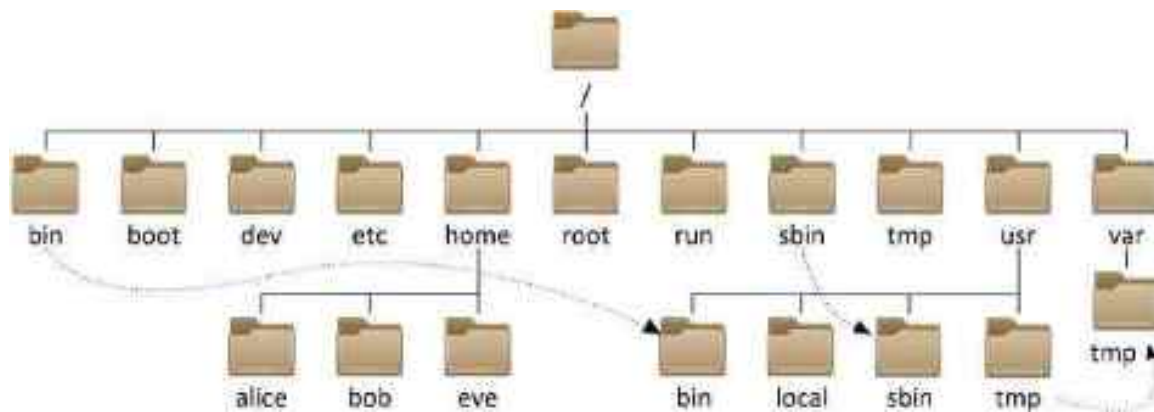
List of student Team Members

- 1.....
- 2.....
- 3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 04: Execute file and Directory manipulation commands.

- I. Practical Significance:** Data and programs are stored in files. These are organized in directories. In a simple way, a directory is just a file that contains other files (or directories). Various operations that can be performed on a file are create, open, read, write, move, close and rename.
- II. Industry / Employer Expected Outcome:**
- 1.To understand the basics of Linux operating system fundamentals and its open-source nature.
 - 2.Basic Scripting Skills for automating tasks and creating custom shell scripts.
 - 3.Ability to perform file operations and manipulate directories.
- III. Course Level Learning Outcomes (COs):**
CO3 - Manage files and directories in Linux operating system.
- IV. Laboratory Learning Outcome:**
Execute the following file and Directory manipulation commands along with different options.
1)pwd 2)cd 3)mkdir 4)rmdir 5)ls 6)cat 7)rm 8)mv 9)cp
- V. Relevant Affective Domain related outcome(s)**
1. Follow precautionary measures.
 2. Follow naming conventions
 3. Follow ethical practices.
- VI. Relevant Theoretical Background**
- Linux uses a hierarchical structure for organizing files and directories. This structure is called as a directory tree. The tree has a single root node, the slash character (/), and all other directories are contained below it. When user first log in to the Unix server, the specified directory is called as Home directory.
- Directory Structure in Unix/Linux:



1. **pwd** : pwd stands for Present Working Directory. This is most used Linux command to see the specific Unix Directory on which the user is working on.

Syntax : \$pwd

2. **cd** : The 'cd' command is used to change directory. You can use it to change to any directory by specifying a valid absolute or relative path.

The syntax is as given below

\$cd <directory name>

Example: \$ cd Directory name

\$cd CO2K

\$cd .. To come out from current working directory.

Example: \$cd ..

\$ cd / It changes to root directory.

Example: \$ cd /

3. **mkdir** : It is used to create a new directory in a current directory.

Syntax: \$mkdir <directory name>

Example: \$mkdir CM5I

User can create more than one directory in a single command.

\$mkdir subject1 subject2

If user wants to create directory on the specific path then syntax is:

Syntax : \$mkdir path/Directory_name

4. **rmdir** :It is used to delete/ remove a directory. If the parent directory having subdirectories then first all subdirectories will be deleted then the parent directory is deleted.

Syntax: **\$rmdir <directory name>**

5. **ls** : It is used to lists files in the current working directory. Syntax:

Syntax: **\$ls [options]**

Options	Meaning
---------	---------

ls -a	list all files including hidden file starting with '.'
-------	--

ls -d	list directories - with ' */'
-------	-------------------------------

ls -i	list file's inode index number
-------	--------------------------------

ls -l	list with long format - show permissions
-------	--

Example:

\$ls -i abc.txt	list contents of abc.txt file
-----------------	-------------------------------

\$ls -l	List of all files along with permissions given to it.
---------	---

\$ls ???	List the names of files with exact three characters in it name.
----------	---

\$ls d*s	List of files names which starts for d and ends with s
----------	--

\$ls -l	Gives full listing of all files and directories.
---------	--

6. **cat**: It is used to create the file and displaying the contents of file/files.

Syntax: **\$ cat >sample** (To create a file)

This is sample file in Unix.

<cntrl d>

\$ cat sample

where cat is a command and sample is argument. This is sample file in Linux. User can display contents of more than one file and called as concatenation.

\$ cat list1.txt list2.txt (List two files at a time)

7. **rm** :It is used to remove the file

Syntax: **\$rm filename**

Example

\$rm abc.txt	Removes abc.txt
--------------	-----------------

\$rm abc.txt xyz.txt Removes abc.txt and xyz.txt files

\$rm * Removes all files

8. **mv** : It is used to move a file from one location to another.

Syntax: mv [Option] source destination

(consider 3 files having name abc.txt, xyz.txt and pqr.txt.)

To rename the file abc.txt to aaa.txt (not exist):

\$ ls

abc.txt xyz.txt pqr.txt

\$mv abc.txt aaa.txt

\$ ls

xyz.txt pqr.txt aaa.txt

Option: -i (Interactive): The mv command with -i option ask for confirmation

\$ mv -i abc.txt aab.txt

9. **cp**: It is used to copies a file to destination file.

If the command contains two file names, then it copies the contents of 1st file to the 2nd file. If the 2nd file doesn't exist, then first it creates one and content is copied to it. But if it existed then it is simply overwritten without any warning.

Syntax: \$cp Source file Destination file

Example:

\$ ls

xyz.txt

\$ cp xyz.txt zzz.txt

Check it now

\$ ls

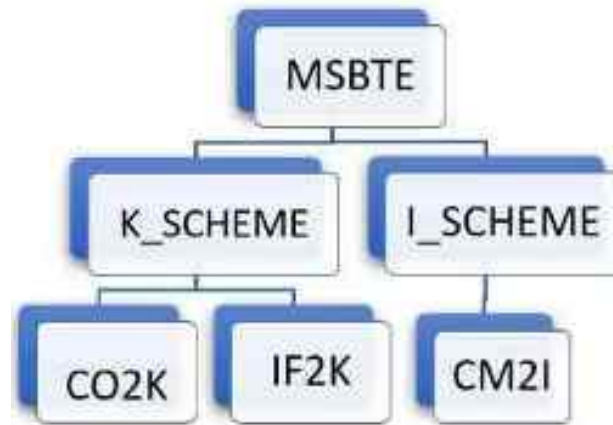
xyz.txt zzz.txt

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

1. Create the following Structure



2. Create 3 files p1 p2 p3

IX. Result(s)

Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. How to shift from Root directory to User (Home) directory?
2. How to see directories?
3. What are different options of ls command? Write down the command along with the option and note down the output. (Use \$man ls command to check options)
4. What are different options of mv command?

(Space for answer)

[illegible]

1.

Exercise:

- . Write the command for performing the following tasks sequentially
 - a. Display your current directory.
 - b. Create a directory 'subject' in the current directory.
 1. Create a file 'sample' in the directory 'subject'.
 - c. Create a file ABCD.txt, create a copy with XXX.txt. Rename the original file with AACD.txt. Delete the file XXX.txt.
 - d. Display the inodes of any two files at the same time.
 - e. Create two files unit1 and unit2 and perform the following operations
 - 1.Copy contents of unit1 to unit2.
 2. Display inodes of two files.
 3. Rename the file 'unit1' to 'Lesson1'

(Space for answer)

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

XIV. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>

2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

XV. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
- 2.....
- 3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 05: Execute file and Directory manipulation commands.

- I. Practical Significance:** Data and programs are stored in files. These are organized in directories. In a simple way, a directory is just a file that contains other files (or directories). Various operations that can be performed on a file are create, open, read, write, move, close and rename.
- II. Industry / Employer Expected Outcome:**
- 1.To understand the basics of Linux operating system fundamentals and its open-source nature.
 - 2.Basic Scripting Skills for automating tasks and creating custom shell scripts.
 - 3.Ability to perform file operations and manipulate directories.
- III. Course Level Learning Outcomes (COs):**
CO3 - Manage files and directories in Linux operating system.
- IV. Laboratory Learning Outcome:**
Execute the following file and Directory manipulation commands along with different options.
1)touch 2)more 3)lp 4)file 5)wc 6)cmp 7)comm 8)diff 9)split
- V. Relevant Affective Domain related outcome(s)**
1. Follow precautionary measures.
 2. Follow naming conventions
 3. Follow ethical practices.
- VI. Relevant Theoretical Background**

- 1. touch:** It is the easiest way to create new, empty files.

Syntax: \$touch filename

Example: \$touch aaa

\$touch a1 a2 a3 Create multiple files

- 2. more:** If the file is too large for its contents to fit in one screen, it will scroll off your screen when the file is displayed with cat command then use more command.

Syntax:\$more ["Space bar" /f/b/j/k/200G/G] filename

Options	Meaning
---------	---------

[Space]	scrolls the display, one screenful of data at a time	j	scrolls forward one line
---------	--	---	--------------------------

f	scrolls forward one screen
---	----------------------------

b	scrolls backwards one screen
---	------------------------------

k scrolls backward one line
200G Goes to line number 200
G Goes to last line of the file

3. **lp:** The lp command arranges for the files specified by the Files parameter and their associated information (called a request) to be printed by a line printer.

If you don't have the lp command installed, you can use the apt package manager to install it.

sudo apt install lprng

syntax :lp -d [printer-id] [filename]

Replace [filename] with the path to the target file.

4. **file:** file command is used to determine the type of a file. file type may be of human-readable(e.g. 'ASCII text') or MIME type(e.g. 'text/plain; charset=us-ascii'). This command tests each argument in an attempt to categorize it.

It has three sets of tests as follows:

1.filesystem test: This test is based on the result which returns from a stat system call. The program verifies that if the file is empty, or if it's some sort of special file. This test causes the file type to be printed.

2.magic test: These tests are used to check for files with data in particular fixed formats.

3.language test: This test search for particular strings which can appear anywhere in the first few blocks of a file.

Syntax: file [option] [filename]

5. **wc:** The wc (word count) command in Unix/Linux operating systems is used to find out number of lines, words and characters in a file.

The syntax of wc command as shown below:

\$wc [-l -w -c] <filename>

-l counts only number of lines

-w counts only number of words

-c counts only number of characters Example:

\$cat student1

Ritu

Meera

Rajan

\$wc student1 3 3 14

Means 3 lines, 3 words and 14 characters.

6. **cmp**: This command is used to compare files when used without options. It uses two filenames as an argument and display the differences on the terminals.

```
$cmp filename1 filename2
```

```
$cat student1          $cat student2
```

```
Harsh                  Harshu
```

```
Sujay                  Sujay
```

```
Smith                  Smith
```

```
$cmp student1 student2
```

```
Student1 student2 differ : char 6 line1
```

7. **comm**: This command compares two sorted files. It compares each line of first file with its corresponding line in the second file and generates three column output.

The first column lists the lines only in first file.

The second column lists the lines only in second file. The third column lists the lines in both files.

Syntax: \$comm filename1 filename2

Example:

```
$cat student1          $cat student2
```

```
aa                    aa
```

```
bb                    pp
```

```
cc                    qq
```

```
$comm student1 student2
```

```
bb pp      aa
```

```
cc qq
```

8. **diff**: This command is used to show difference between two text files. It also tells which line in one has to be changed to make the two files identical.

Syntax: \$diff filename1 filename2

The options of the result should be like this –

a -Added the text to file

c -Changes are made in the file

d -Deletion operation is performed

< -Lines from the first file

> -Lines from the second file

\$cat file1.txt

I need to go to the shop.

I need to buy some mangoes. When I get home, I'll wash the cat.

\$cat file2.txt

I need to go to the shop.

I need to buy some mangoes.

Oh yeah, I also need to buy cheese. When I get home, I'll wash the cat.

Use the diff command to compare both files.

\$ diff file1.txt file2.txt

The above command should give the result as shown below –

2a3

> Oh yeah, I also need to buy cheese.

From the output, 2a3 means “After line 2 in the first file, a line needs to be added: line 3 from the second file”.

9. **split:** It is used to split the large file into smaller files. Default size is 1000 lines per file.

Syntax: \$split -n filename tag name

-n number of lines in each smaller file.

-tag name by default the split builds the output files named xaa,xab,xac.....

If tag name specified, it replaces the x with that tag name. Example:

Split -10 student

Check it now \$ls

xaaxab xac

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

1. Create 3 blank files p1 p2 p3 using touch command

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. What are options of wc command
2. What is difference between cat pqr and cat pqr|more(file pqr consists of more than 25 lines in it)
3. What are different options of diff command?
4. What is difference between comm and cmp command

(Space for answer)

XI. Exercise:

1. Write the commands for:
 - a) Counting number of words in the 'data.txt'
 - b) Counting number of lines in 'data.txt'
 - c) Counting all characters in the 'data.txt'

(Space for answer)

XII. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 06: Execute Linux commands for compressing, decompressing, and archiving files.

I. **Practical Significance:** Linux file compression commands reduce the size of files and directories by compressing them, so they are easier to store and transfer. Multiple files and directories can be grouped and stored as a single archive file with archiving commands.

II. **INDUSTRY / EMPLOYER EXPECTED OUTCOME:**

- 1) To understand the basics of Linux operating system fundamentals and its open- source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. **COURSE LEVEL LEARNING OUTCOMES (COS):**

CO3 - Manage files and directories in Linux operating system.

IV. **LABORATORY LEARNING OUTCOME:**

Execute Linux commands for compressing, decompressing, and archiving files.

V. **Relevant Affective Domain related outcome(s)**

1. Follow precautionary measures.
2. Follow installation steps.
3. Follow ethical practices.

VI. **Relevant Theoretical Background**

1. **gzip:**gzip, short for GNU Zip, is a command-line compression tool commonly found on Linux systems. It utilizes the DEFLATE compression algorithm to reduce the size of files, making them more manageable for storage and transmission.

Syntax of the gzip Command: `gzip [Options] [filenames]`

This syntax allows users to compress a specified file.

Options Available in gzip Command

Sr.No	Options	Description
1	-f	Forcefully compress a file even if a compressed version with the same name already exists.
2	-k	Compress a file and keep the original file, resulting in both the compressed and original files.
3	-L	Display the gzip license for the software.

4	-r	Recursively compress all files in a folder and its subfolders.
5	-v	Display the name and percentage reduction for each file compressed or decompressed.
6	-d	Decompress a file that was compressed using the gzip command.

To compress a file named “mydoc.txt,” the following command can be used:

Example:

```
gzip mydoc.txt
```

This command will create a compressed file of mydoc.txt named as mydoc.txt.gz and delete the original file.

2. **gunzip:** gunzip command is used to compress or expand a file or a list of files in Linux. It accepts all the files having extension as **.gz, .z, _z, -gz, -z, .Z, .taz** or **.tgz** and replace the compressed file with the original file by default. The files after uncompression retain its actual extension.

Syntax:

```
gunzip [Option] [archive name/file name]
```

Example 1: The argument that is passed here is: **geeksforgeeks.txt** which is a compressed text file.

Input:

```
Sc@ubuntu:gunzip abc.txt
```

Output:

```
abc.txt.gz
```

Example 2: The argument that is passed here is: **abc.txt.gz** which is compressed file

Input:

```
Sc@ubuntu:gunzip abc.txt.gz
```

Output:

```
abc.txt
```

If a file is compressed using *gzip* command, a suffix i.e. **.gz** will be added to the file name after compression. Hence while uncompressing this file we can either use the original file name as shown in *Example 1* or the filename with the suffix **.gz** as shown in *Example 2* as an argument.

Example 3: In order to uncompress multiple files using the *gunzip* command, we can pass multiple file names as an argument as shown in the below example:

Syntax:

```
gunzip [file1] [file2] [file3]...
```

Input:

```
Sc@ubuntu:gunzip abc.txt.gz gfg.txt
```

Output:

```
abc.txt, gfg.txt
```

3.tar : The Linux ‘tar’ stands for tape archive, which is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands that provides archiving functionality in Linux. We can use the Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Syntax :

```
tar [options] [archive-file] [file or directory to be archived]
```

Here ,

- **tar:** The command itself.
- **[options]:** Optional flags or settings that modify the behavior of the tar command.
- **[archive-file]:** The name of the archive file you are creating or working with.
- **[file or directory to be archived]:** The file or directory you want to include in the archive.

Sr.No	Options	Description
1	-c	Creates an archive by bundling files and directories together.
2	-x	Extracts files and directories from an existing archive.
3	-f	Specifies the filename of the archive to be created or extracted.
4	-t	Displays or lists the files and directories contained within an archive.
5	-u	Archives and adds new files or directories to an existing archive.
6	-v	Displays verbose information, providing detailed output during the archiving or extraction process.
7	-A	Concatenates multiple archive files into a single archive.
8	-z	Uses gzip compression when creating a tar file, resulting in a compressed archive with the ‘.tar.gz’ extension.
9	-j	Uses bzip2 compression when creating a tar file, resulting in a compressed archive with the ‘.tar.bz2’ extension.

10	-W	Verifies the integrity of an archive file, ensuring its contents are not corrupted.
11	-r	Updates or adds files or directories to an already existing archive without recreating the entire archive.

4. **zip**: ZIP is a compression and file packaging utility for [Unix](#). Each file is stored in a single .zip { .zip-filename } file with the extension .zip.

syntax :

zip [file_name.zip] [file_name]

5. **unzip**:unzip will list, test, or extract files from a ZIP archive, commonly found on Unix systems. The default behavior (with no options) is to extract into the current directory (and sub-directories below it) all files from the specified ZIP archive.

Syntax:

unzip [file_name.zip]

Example:

Suppose we have a zip file “name = jayesh_gfg.zip” and we have three text files inside it “name = a.txt, b.txt and c.txt”. we have to unzip it in the current directory.

Syntax :

unzip jayesh_gfg.zip

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure

Teacher must assign a separate program statement to students.

Note:Write terminal output in Result Section

IX. Result(s)

.....

.....

.....

.....

.....

.....

X. Practical related questions(Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

- 1) create and extract a zip file in linux?
- 2) What is the command to decompress all files with a .gz extension in the current directory?
- 3) Explain gunzip command with all options?

(Space for answer)

[illegible]

Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix/>

2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>

X. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
- 2.....
- 3.....
- 4.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 07: Change file and directory permissions.

- 1) ls -l, ls -ld
- 2) chmod (with all options)
- 3) chown
- 4) chgrp

I. Practical Significance:

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning outcome(s)

CO2 - Execute general purpose commands of the Linux operating system.

IV. Laboratory Learning outcome(s)

Execute the commands to change file and directory permissions.

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions.
3. Follow ethical practices.

VI. Relevant Theoretical Background (with diagrams if required)

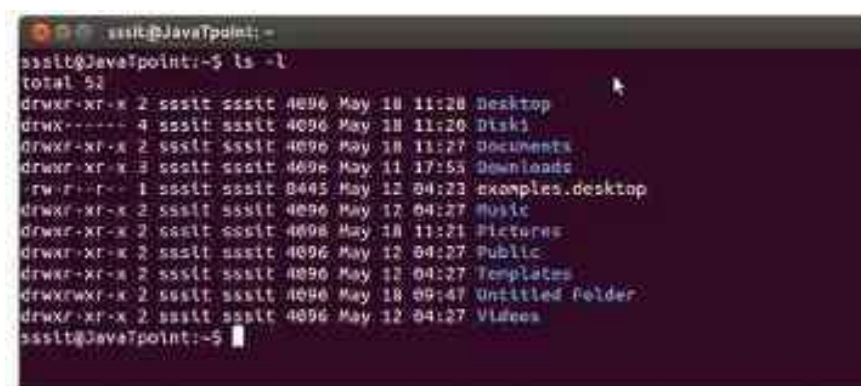
Command to change file directory and permission:

1. **Ls command:** The **ls** is the list command in Linux. It will show the full list or content of your directory. Just type **ls** and press the enter key. The whole content will be shown.

ls -l :it displays various information related to file permission. **-l**: It stands for long format. It shows Unix file types, number of hard links, permissions, group, owner, last modified name and date-time, and size. If the changed date is older than six months, the time is substituted with the year. A few implementations add extra flags to permissions

Syntax: ls options

Example :ls-l



```
ssst@JavaPoint: ~$ ls -l
total 52
drwxr-xr-x 2 ssst ssst 4096 May 18 11:28 Desktop
drwxr-xr-x 4 ssst ssst 4096 May 18 11:28 Disk1
drwxr-xr-x 2 ssst ssst 4096 May 18 11:27 Documents
drwxr-xr-x 3 ssst ssst 4096 May 11 17:53 Downloads
-rw-r--r-- 1 ssst ssst 8445 May 12 04:23 examples.desktop
drwxr-xr-x 2 ssst ssst 4096 May 12 04:27 Music
drwxr-xr-x 2 ssst ssst 4096 May 18 11:21 Pictures
drwxr-xr-x 2 ssst ssst 4096 May 12 04:27 Public
drwxr-xr-x 2 ssst ssst 4096 May 12 04:27 Templates
drwxr-xr-x 2 ssst ssst 4096 May 18 09:47 Untitled Folder
drwxr-xr-x 2 ssst ssst 4096 May 12 04:27 Videos
ssst@JavaPoint: ~$
```

1. Column 1 indicates information regarding file permission.
2. Column 2 indicates the number of links to the file.
3. Column 3 & 4 indicates the owner and group information.
4. Column 5 indicates size of the file in bytes.
5. Column 6 shows the date and time on which the file was recently modified.
6. Column 7 shows the file or directory name

Ls-ld: Sometimes you don't want to see the files within the directory; you just want to see just the permissions on the directory itself. For that you should specify the `-d` command-line option as well. With the `ls -ld` command, you'll see the permissions on the directory itself, not on its contents.

- `ls -ld ~/tmp`
- `drwxr-x--x 2 lmui 512 Aug 25 17:32 /home/lmui/tmp`

This tells you that the directory `tmp` in my home directory is read/write to myself and read-only for my group. It cannot be read or written by anyone else; to the directory, since it is executable for all. The initial `d` means that it's a directory.

2. **chmod:**

Linux `chmod` command is used to change the access permissions of files and directories. It stands for **change mode**.

Syntax: `chmod <options> <permissions> <file name>`

The `chmod` command supports the following command-line options:

1. **-c, --changes:** It is similar to the verbose option, but the difference is that it is reported if a change has been made.
2. **-f, --silent, --quiet:** It is used to suppress the error messages.
3. **-v, --verbose:** It is used to display a diagnostic for every processed file.
4. **--no-preserve-root:** It is used for not treating the backslash symbol ('/'), especially (the default).
5. **--preserve-root:** If this option is used, it will fail to operate recursively on backslash ('/').
6. **--reference=RFILE:** It is used to specify the RFILE's mode alternatively MODE values.
7. **-R, --recursive:** It is used to change files and directories recursively.
8. **--help:** It is used to display the help manual having a brief description of usage and support options.
9. **--version:** It is used to display the version information.

3. Chown:

The **`chown`** command, short for “change owner,” is a powerful tool that allows users to change the owner of files and directories. This command is particularly useful in scenarios where administrators need to grant or revoke access to specific resources.

Syntax of chown Command in Linux

The chown command in Linux has the following syntax:

Syntax: chown [options] new_owner[:new_group] file(s)

- **`chown`:** The base command.
- **`options`:** Optional flags that modify the behavior of the **`chown`** command.
- **`new\_owner[:new\_group]`:** The new owner and optionally the new group. If **`new\_group`** is omitted, only the owner is changed.
- **`file(s)`:** The file or files for which ownership is to be changed.

4. Chgrp:

The **`chgrp`** command in Linux is used to change the group ownership of a file or directory.

Syntax of `chgrp` command in Linux

chgrp [OPTION]... GROUP FILE...

chgrp [OPTION]... --reference=RFILE FILE...

Examples of `chgrp` command in Linux **Changing Group Ownership of a Single File**

To change the group ownership of a file. `sudo chgrp geeksforgeeks abc.txt`

```

kcvirtual@kcvirtual-VirtualBox:~/GFG$ ls -l
total 4
-rw-r--r-- 1 kcvirtual kcvirtual 8 Apr 13 06:39 abc.txt
kcvirtual@kcvirtual-VirtualBox:~/GFG$ sudo chgrp geeksforgeeks abc.txt
kcvirtual@kcvirtual-VirtualBox:~/GFG$ ls -l
total 4
-rw-r--r-- 1 kcvirtual geeksforgeeks 8 Apr 13 06:39 abc.txt
kcvirtual@kcvirtual-VirtualBox:~/GFG$

```

Here the group name of the file *abc.txt* was changed from *kcVirtual* to *geeksforgeeks*. Note that when files are created the groupname of the file is the same as the owner under which the file was created.

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure: Teacher must assign a separate command statements to students

List down all options of *ls* command.

Note:Write terminal output in Result Section

IX. Result(s)

.....

.....

.....

.....

.....

X. Practical related questions (Provide space for answers)

- 1.explain the differences and uses of *chmod*, *chown* and *chgrp* commands in Linux?
- 2.what is meant by ownership permissions and how they function in an operating system?
3. How would you manage permissions in a distributed file system?
4. Explisn how to change file/directory permission using shorthand notations?

XI. References/Suggestions for further reading:

<https://www.computerhope.com/unix/uchmod.htm>

<https://www.geeksforgeeks.org/>

XI. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

1.....

2.....

3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 08: Use the vi editor to create and edit files.**I. Practical Significance:**

There are many ways to edit files in Unix. Editing files using the screen-oriented text editor vi is one of the best ways. This editor enables you to edit lines in context with other lines in the file. Vi is intended as a plain text editor (similar to Notepad on Windows, or Textedit on Mac) as opposed to a word processing suite such as Word or Pages. It does, however have a lot more power compared to Notepad or Textedit.

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning outcome(s)

CO4 - Use vi editor in Linux operating system.

IV. LABORATORY LEARNING OUTCOME:

Use vi editor and execute all editor commands

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions.
3. Follow ethical practices.

VI. Relevant Theoretical Background

The following table lists out the basic commands to use the vi editor –

Sr.NO	Command	Description
1	vi filename	Creates a new file if it already does not exist, otherwise opens an existing file.
2	vi -R filename	Opens an existing file in the read-only mode.
3	view filename	Opens an existing file in the read-only mode.

Following is an example to create a new file test1 if it already does not exist in the current working directory – The above command will generate the following output

```
$vi test1
```

```
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

```
"test1" [New File]
```

You will notice a tilde (~) on each line following the cursor. A tilde represents an unused line. If a line does not begin with a tilde and appears to be blank, there is a space, tab, newline, or some other non-viewable character present.

Operation Modes

While working with the vi editor, we usually come across the following two modes –

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

Insert mode – This mode enables you to insert text into the file. Everything that's typed in Operating System (22516) Maharashtra State Board of Technical Education 58 this mode is interpreted as input and placed in the file. vi always starts in the command mode. To enter text, you must be in the insert mode for which simply type i. To come out of the insert mode, press the Esc key, which will take you back to the command mode.

Moving within a File

To move around within a file without affecting your text, you must be in the command mode (press Esc twice).

Sr.NO	Command	Description
-------	---------	-------------

1	k	Moves the cursor up one line
2	j	Moves the cursor down one line
3	h	Moves the cursor to the left one character position
4	l	Moves the cursor to the right one character position

Control Commands:**Editing Files**

To edit the file, you need to be in the insert mode. There are many ways to enter the insert mode from the command mode –

Sr.NO	Command	Description
1	i	Inserts text before the current cursor location
2	I	Inserts text at the beginning of the current line
3	a	Inserts text after the current cursor location
4	A	Inserts text at the end of the current line
5	o	Creates a new line for text entry below the cursor location
6	O	Creates a new line for text entry above the cursor location

Page movement Description:

Sr.No.	Command	Description
1	CTRL+d-	Move forward 1/2 screen
2	CTRL+f-	Move forward one full screen
3	CTRL+u -	Move backward 1/2 screen
4	CTRL+b	Move backward one full screen
5	CTRL+e	Moves screen up one line
6	CTRL+y-	Moves screen down one line
7	CTRL+I	Redraws screen

Deleting Characters:

Here is a list of important commands, which can be used to delete characters and lines in an open file –

Sr.No.	Command	Description
1	x	Deletes the character under the cursor location
2	X	Deletes the character before the cursor location
3	Dw	Deletes from the current cursor location to the next word
4	d^	Deletes from the current cursor position to the beginning of the line
5	d\$	Deletes from the current cursor position to the end of the line
6	D	Deletes from the cursor position to the end of the current line
7	dd	Deletes the line the cursor is on

Change Commands

You also have the capability to change characters, words, or lines in vi without deleting them.

Sr.No.	Command	Description
1	cc	Removes the contents of the line, leaving you in insert mode.
2	cw	Changes the word the cursor is on from the cursor to the lowercase w end of the word.
3	r	Replaces the character under the cursor. vi returns to the command mode after the replacement is entered.
4	R	Overwrites multiple characters beginning with the character currently under the cursor. You must use Esc to stop the overwriting.
5	s	Replaces the current character with the character you type. Afterward, you are left in the insert mode.
6	S	Deletes the line the cursor is on and replaces it with the new text. After the new text is entered, vi remains in the insert mode.

Copy and Paste Commands

You can copy lines or words from one place and then you can paste them at another place using the following commands –

Sr.No.	Command	Description
1	yy	Copies the current line.
2	yw-	Copies the current word from the character the lowercase w cursor is on, until the end of the word.
3	p	Puts the copied text after the cursor
4	P	Puts the yanked text before the cursor.

Ex mode command

Press Esc key and then (:) colon to enter ex-mode commands. A colon is displayed at the lefthand corner of the last line on your screen.

Command	Action
:w	Saves file and remains in editing mode

:x	Saves file and quits editing mode
:wq	Save and quit
:q	Quits vi when no changes are made
:q!	Quits vi cancelling the changes
:sh	Escape to Unix shell

Advanced Commands:

There are some advanced commands that simplify day-to-day editing and allow for more efficient use of vi –

Word and Character Searching:

The vi editor has two kinds of searches: **string** and **character**. For a string search, the / and ? commands are used. When you start these commands, the command just typed will be shown on the last line of the screen, where you type the particular string to look for.

These two commands differ only in the direction where the search takes place –

- The / command searches forwards (downwards) in the file.
- The ? command searches backwards (upwards) in the file.

The **n** and **N** commands repeat the previous search command in the same or the opposite direction, respectively. Some characters have special meanings. These characters must be preceded by a backslash (\) to be included as part of the search expression.

Sr.No.	Character	Description
1	^	Searches at the beginning of the line (Use at the beginning of a search expression).
2	.	Matches a single character.
3	*	Matches zero or more of the previous character.
4	\$	End of the line (Use at the end of the search expression).
5	[	Starts a set of matching or non-matching expressions.
6	<	This is put in an expression escaped with the backslash to find the ending or the beginning of a word.
7	>	This helps see the '<' character description above.

Running Commands:

The vi has the capability to run commands from within the editor. To run a command, you only need to go to the command mode and type **:! command**. For example, if you want to check whether a file exists before you try to save your file with that filename, you can type **:! ls** and you will see the output of **ls** on the screen. You can press any key (or the command's escape sequence) to return to your vi session.

Replacing Text:

The substitution command (**:s/**) enables you to quickly replace words or groups of words within your files. Following is the syntax to replace text –

:s/search/replace/g

The **g** stands for globally. The result of this command is that all occurrences on the cursor's line are changed.

VII. Required Resources/apparatus/equipment with specifications

No.	Equipment Name with Broad Specifications	Relevant LLO Number
	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and opensource operating System. (RedHat, Ubuntu etc.).	

VIII. Procedure**I. Teacher must assign a separate program statement to students.**

Create a file OS and type at least 30 lines in it.

- Write command to save the file
- Write the command to insert text at the end of line.
- Replace some character into beginning of second line.
- Write a command to delete 10 lines at a time.
- Write a command to move cursor 3 word to the right

IX. Result(s)

X. Practical related questions (Provide space for answers)

1. What is the difference between Type o and Type O?
2. State the three ways quit the vi-editor.
3. What is effect of (Type this at colon mode and observe the changes in your file)
 - a. set nu
 - b. set ic
 - c. set nonu
 - d. set noai
4. Give vi command to delete line to the left of cursor.
5. What is difference between yank and delete?
6. What are different modes in Vi editor?
7. Observe the output of following commands:

i) at colon mode

:abbr MSBTE Maharashtra State Board Of Technical
Education Now in the beginning of your file type MSBTE
and press space barWrite down the output

ii) Go to beginning of your file press
4YY Then move to end of your file
and press 'P'

Write the purpose of the command

(Space for answer)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References/ Suggestions for Further Reading

1. <https://www.tutorialspoint.com/unix>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>
4. <https://www.openvim.com/tutorial.html>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

1.....

2.....

3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 09: Use wildcard characters..

- I. **Practical Significance:** A wildcard is a symbol that takes the place of an unknown character or set of characters. Commonly used wildcards are the asterisk (*) and the question mark (?). Depending on the software or the search engine you are using, other wildcard characters may be defined.

When you are searching for files in UNIX, DOS, or Windows, or on the web, you can simplify your search by using a wildcard. Wildcards may also simplify commands issued from the command line in UNIX or DOS.

II. **INDUSTRY / EMPLOYER EXPECTED OUTCOME:**

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. **COURSE LEVEL LEARNING OUTCOMES (COS):**

CO4 - Use vi editor in Linux operating system.

IV. **LABORATORY LEARNING OUTCOME:**

Use wildcard characters (e.g., *, ?, []) to list and manipulate specific sets of files within the directory.

V. **Relevant Affective Domain related outcome(s)**

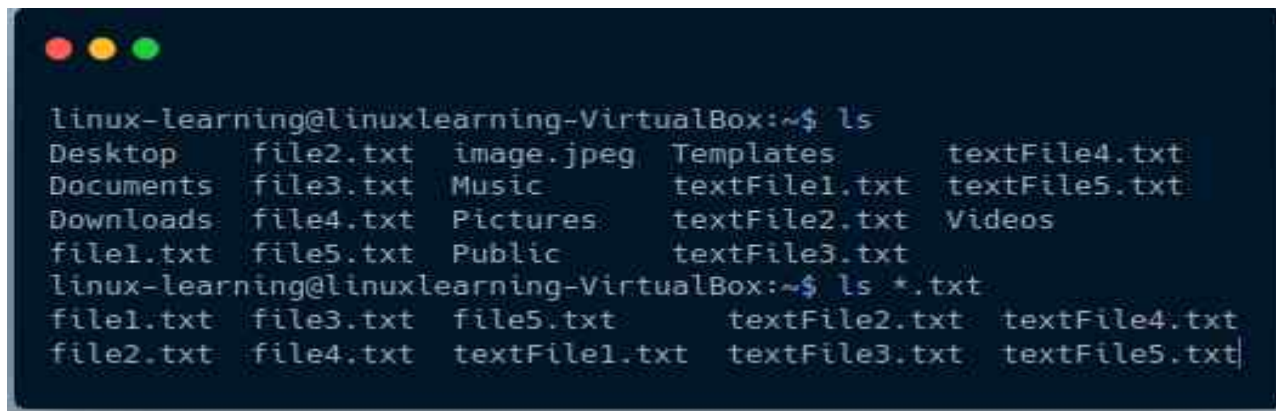
1. Follow precautionary measures.
2. Follow installation steps.
3. Follow ethical practices.

VI. **Relevant Theoretical Background:**

Since the Linux system uses filename so much it provides us with special characters to help us rapidly specify groups of filenames. These special characters are called **wildcards**.

Sr. NO	Wildcard	Meaning
	*	Matches any character
	?	Matches any single character
	[]	Matches any character that is a member of the set character

1. List all the files **ending** with **.txt**

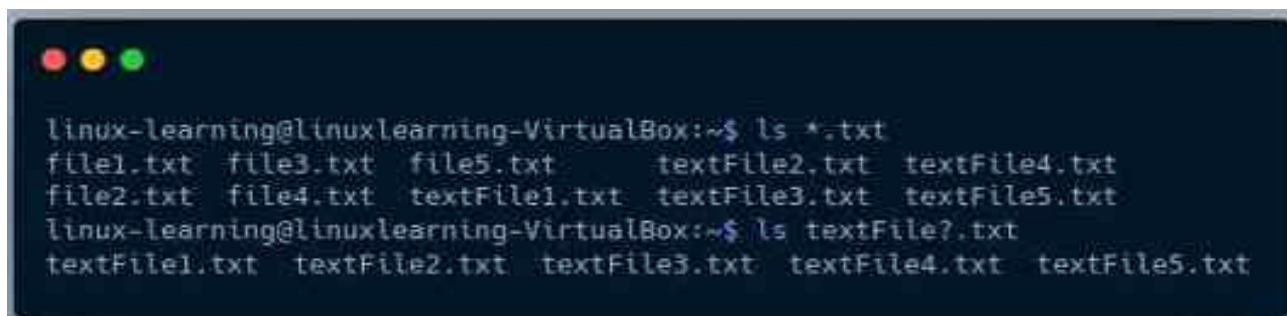


```
linux-learning@linuxlearning-VirtualBox:~$ ls
Desktop  file2.txt  image.jpeg  Templates  textFile4.txt
Documents file3.txt  Music       textFile1.txt textFile5.txt
Downloads file4.txt  Pictures    textFile2.txt Videos
file1.txt file5.txt  Public      textFile3.txt
linux-learning@linuxlearning-VirtualBox:~$ ls *.txt
file1.txt  file3.txt  file5.txt  textFile2.txt  textFile4.txt
file2.txt  file4.txt  textFile1.txt textFile3.txt  textFile5.txt
```

In the above example, we first use the `ls` command to list the directory content and then since we only wanted the files that end with `.txt` we used wildcard `*` and created the command `ls *.txt`

`ls *.txt` where
`*` matches any character and
`.txt` means any file that ends with **.txt**.

2. List all the `.txt` files that begin with "textFile" and have exactly 1 character after that. For example **textFile9.txt**



```
linux-learning@linuxlearning-VirtualBox:~$ ls *.txt
file1.txt  file3.txt  file5.txt  textFile2.txt  textFile4.txt
file2.txt  file4.txt  textFile1.txt textFile3.txt  textFile5.txt
linux-learning@linuxlearning-VirtualBox:~$ ls textFile?.txt
textFile1.txt  textFile2.txt  textFile3.txt  textFile4.txt  textFile5.txt
```

In the above example, we first listed all the text files in the directory using `ls *.txt` and then we used wildcard `?` and created command `ls textFile?.txt` to filter the results as per our requirement

So let's break down the command

`ls textFile?.txt`

where

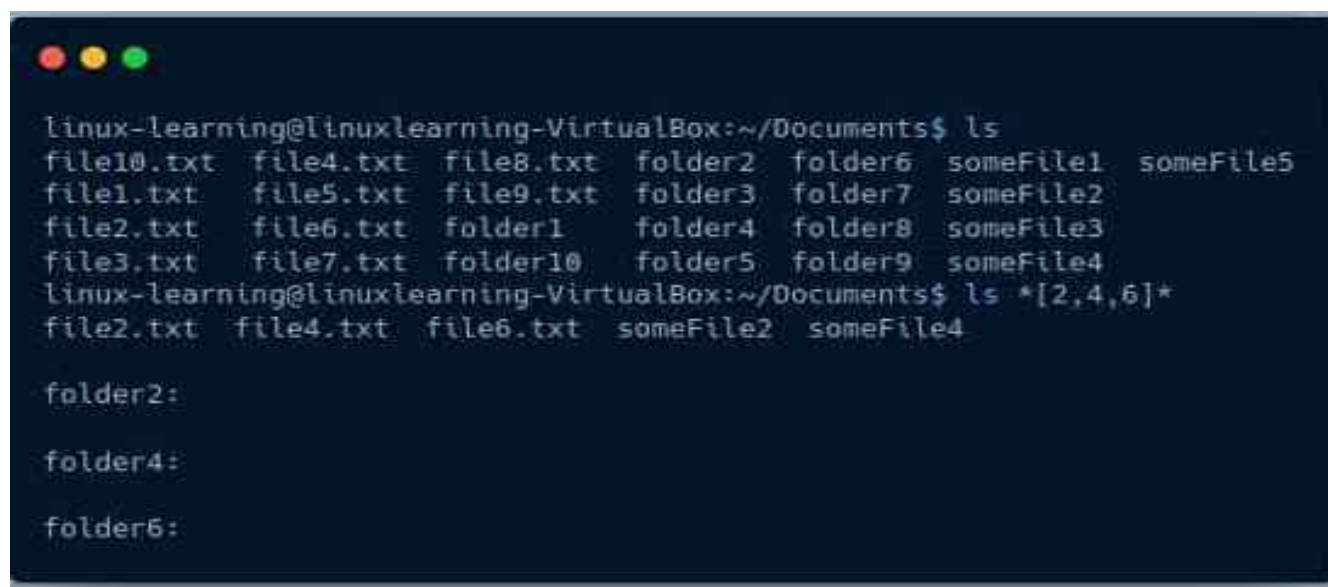
`textFile` specifies that files should begin with "textFile"

One `?` after the `textFile` since we wanted files that have exactly one character after "textFile"

`.txt` means any file that has **.txt** in the end.

3. What if we wanted to get only the **files and folders** that have **numbers 2, 4, or 6**.

So the file can start with any character and end with any character but it should contain at least one of the three numbers



```
linux-learning@linuxlearning-VirtualBox:~/Documents$ ls
file10.txt  file4.txt  file8.txt  folder2  folder6  someFile1  someFile5
file1.txt   file5.txt  file9.txt  folder3  folder7  someFile2
file2.txt   file6.txt  folder1    folder4  folder8  someFile3
file3.txt   file7.txt  folder10   folder5  folder9  someFile4
linux-learning@linuxlearning-VirtualBox:~/Documents$ ls *[2,4,6]*
file2.txt  file4.txt  file6.txt  someFile2  someFile4

folder2:

folder4:

folder6:
```

In the above example, we used the wildcard `[]` and created a command `ls *[2,4,6]*` to get the desired results.

So let's break down the command

`ls *[2,4,6]*` where

`*` represents any character

`[2,4,6]` represent that the file and folders with either number 2 or 4 or 6

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure

Explain all wild card characters.

Note: Write terminal output in Result Section

IX. Result(s)

.....

.....

.....

.....

X. Practical related questions

(Provide space for answers)

1. List all files with a ".txt" extension in the current directory:
2. Remove all files with a ".bak" extension in the current directory:
3. Display the content of all files starting with "log" and ending with ".txt":
4. Move all files with a ".jpg" extension from the current directory to the "images" directory:
5. Count the number of files with names starting with "file" and ending with a numeric value:

[illegible]

XI. References/Suggestions for further reading: include websites/links

<https://www.geeksforgeeks.org/>

<https://stackoverflow.com/>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 10: Use of Pipes in Linux.**I. Practical Significance:**

Pipes are a powerful and essential concept in the Linux command-line environment. They allow you to connect the output of one command as the input to another, enabling you to create complex and efficient data processing workflows.

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. COURSE LEVEL LEARNING OUTCOMES (COS):

CO4 - Use vi editor in Linux operating system.

IV. LABORATORY LEARNING OUTCOME:

- a) Create a text file with various lines of text.
- b) Create a complex pipeline by chaining multiple commands together using pipes (|).

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow installation steps.
3. Follow ethical practices.

VI. Relevant Theoretical Background

- a) Create a text file with various lines of text.

1) touch command

This is the most standard command to quickly create an empty text file. The commands are as follows:

Syntax: touch filename.txt

type the word touch followed by the name of the file you like to give it, you have created an empty text file inside of a terminal. You can add the file names of the file you would like to create at once with space in between each filename. The command below creates three empty files at once using the touch command and you can create as many files as you like.

Example: touch file1.txt file2.txt file3.txt

2) Standard Redirect Symbol (>)

Creating a text file in the terminal is a straightforward process that requires minimal effort. It works efficiently when creating a single text file promptly. However, when the need arises to create multiple text files simultaneously, the process can become tedious. The command involves using the standard redirect symbol (>) followed by a spacebar and the desired file name.

Syntax:> filename.txt

If you want to create several text files at once, then you can add the redirect symbol after the previous filename and chain the command repeatedly to create multiple empty files.

Syntax:> file.txt > file2.txt > file3.txt

The above command creates three empty text files. The redirect symbol is quite time-saving if you just want to create a single text file. It gets quite longer than the touch command to create multiple empty text files.

3) CAT Command

This method is incredibly simple and user-friendly. To create a new text file using this method, all you need to do is type “CAT” followed by two redirect symbols (>>) and the desired file name. While it is not necessary to use the >> symbols, you can also use a single > symbol. However, caution must be exercised when using a single > symbol, as it can inadvertently overwrite the existing content in the text file if the file already exists. This method combines the functionality of the touch and redirect symbol commands. It is a unique approach, best suited for creating empty, never-edited files. If you prefer to create and type directly into the text file, this method is highly efficient. It eliminates the need to open a separate editor, saving you time and offering a straightforward command.

The below command creates an empty yet edited file as it prompts the user to create a text file and type in the file at the same time. So, if you do not want to edit the file, simply press CTRL+C and it will simply exit and create an empty file.

Syntax:cat >> file.txt

But, if you would like to add some text to the file, you can type in after this, like this:

cat >> new.txt

4) Using echo / printf

The echo command, similar to the cat command but more versatile, is commonly employed to display text on the terminal. However, its functionality extends beyond that, as it can also be used to write content to a file or create an empty file. To accomplish this, the echo command is utilized in conjunction with double redirect symbols (single “>” can also be used) followed by the desired filename.

Syntax:echo >> filename.txt

If you want to create multiple files at a time, you can chain up the command as in previous methods.

Syntax:echo >> file1.txt >> file2.txt >> file3.txt

Similar to the echo command, we have the printf command as well. The print command does the same thing as the echo command but in a C style rather than shell-style editing.

```
printf "" >> filename.txt
```

```
printf "" >> file1.txt >> file2.txt >> file3.txt
```

```
printf "This is some text here \n The second line \n The third line" >> file.txt
```

The print command does some pretty C-like things, such as the newline character and the variable names can be used as well, but that is not for a simple text file. But still, the printf command can be useful in a lot of cases to edit files on the go.

5) Any command-line text editor(Vim, nano)

This is the most time-consuming method and not the fastest, yet the method can be useful for Linux beginners. If you want to heavily edit a text file, you can use command-line text-editors such as Vim, nano, and there are other options as well. But most people use nano as it is simple to use and quick to go. Vim can also be used but most beginners find it difficult to use, so we'll stick with nano for this example.

```
nano filename.txt
```

```
vim filename.txt
```

b) Create a complex pipeline by chaining multiple commands together using pipes (|).

Example:

Cat logfile.txt|grep"error"|sort|uniq-c|awk '{print \$2,\$1}'>error_count.txt

1. Reads the content of "logfile.txt" using cat.
2. Filters lines containing "error" using grep.
3. Sorts the filtered lines using sort.
4. Removes duplicate consecutive lines and counts occurrences with uniq -c.
5. Uses awk to rearrange the columns and redirect the output to "error_count.txt".

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure

Create a text file and perform all commands.

Note: Write terminal output in Result Section

IX. Result(s)

.....
.....
.....

X. Practical related questions (Provide space for answers)

1. Count the number of lines in a file
2. Search for a specific pattern in a file
3. Sort the lines of a file
4. Combine multiple commands

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XI. References/Suggestions for further reading:

<https://www.geeksforgeeks.org/>

<https://stackoverflow.com/>

Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 11: Execute input and output redirection in Linux.

- I. **Practical Significance: I/O (Input-Output) redirection** in Linux is a feature supported by shells that allow us to redirect the outputs and inputs of commands to files, folders, or even different commands. To redirect Linux I/O streams, we make use of the > and < operators to control the direction of the I/O streams.

II. **INDUSTRY / EMPLOYER EXPECTED OUTCOME:**

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. **COURSE LEVEL LEARNING OUTCOMES (COS):**

CO4 - Use vi editor in Linux operating system.

IV. **LABORATORY LEARNING OUTCOME:**

Create input and output redirection in Linux.

V. **Relevant Affective Domain related outcome(s)**

1. Follow precautionary measures.
2. Follow installation steps.
3. Follow ethical practices.

VI. **Relevant Theoretical Background**

1) **Input Redirection**

Just as the output of a command can be redirected to a file, so can the input of a command be redirected from a file. As the **greater-than character** > is used for output redirection, the **less-than character** < is used to redirect the input of a command.

The commands that normally take their input from the standard input can have their input redirected from a file in this manner. For example, to count the number of lines in the file *users* generated above, you can execute the command as follows –

```
$ wc -l users
```

```
2 users
```

```
$
```

Upon execution, you will receive the following output. You can count the number of lines in the file by redirecting the standard input of the **wc** command from the file *users* –

```
$ wc -l < users
```

2

\$

Note that there is a difference in the output produced by the two forms of the `wc` command. In the first case, the name of the file `users` is listed with the line count; in the second case, it is not.

In the first case, `wc` knows that it is reading its input from the file `users`. In the second case, it only knows that it is reading its input from standard input so it does not display file name.

2) Output Redirection

The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection.

If the notation `> file` is appended to any command that normally writes its output to standard output, the output of that command will be written to file instead of your terminal.

Check the following **who** command which redirects the complete output of the command in the `users` file.

```
$ who > users
```

Notice that no output appears at the terminal. This is because the output has been redirected from the default standard output device (the terminal) into the specified file. You can check the `users` file for the complete content –

```
$ cat users
```

```
oko    tty01  Sep 12 07:30
ai      tty15  Sep 12 13:32
ruth    tty21  Sep 12 10:10
pat     tty24  Sep 12 13:07
steve   tty25  Sep 12 13:03
```

```
$
```

If a command has its output redirected to a file and the file already contains some data, that data will be lost. Consider the following example –

```
$ echo line 1 > users
```

```
$ cat users
```

line 1

\$

You can use >> operator to append the output in an existing file as follows –

\$ echo line 2 >> users

\$ cat users

line 1

line 2

\$

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure

IX. Result(s)

.....
.....
.....
.....
.....
.....

X. Practical related questions (Provide space for answers)

- 1) What is input redirection in Linux
- 2) How can you redirect both standard output and standard error to a file in Linux
- 3) What is the purpose of the >> operator in Linux output redirection?

.....
.....
.....
.....
.....
.....
.....
.....

[illegible]

XI. References/Suggestions for further reading:

<https://www.computerhope.com/unix/uchmod.htm>

<https://www.geeksforgeeks.org/>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

1.....

2.....

3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 12: Execute the following filters commands in Linux.

- I. **Practical Significance:** A filter is a program that takes a flow of data from the standard input, processes or filters it and send the result to standard output.
- II. **Industry / Employer Expected Outcome:**
 - 1) To understand the basics of Linux operating system fundamentals and its open- source nature.
 - 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
 - 3) Ability to perform file operations and manipulate directories.
- III. **Course Level Learning Outcomes (COs):**
CO5 – Write programs using shell script.
- IV. **Laboratory Learning Outcome:**
Execute the following filters command in Linux 1)pr 2)head 3) tail 4) cut 5) paste 6) sort 7)uniq 8)tr
- V. **Relevant Affective Domain related outcome(s)**
 1. Follow precautionary measures.
 2. Follow naming conventions.
 3. Follow ethical practices.
- VI. **Relevant Theoretical Background**
 1. **pr - convert text files for printing**

The pr command does minor formatting of files on the terminal screen or for a printer. For example, if you have a long list of names in a file, you can format it onscreen into two or more columns.

syntax :**pr option(s) filename(s)**

pr changes the format of the file only on the screen or on the printed copy; it doesn't modify the original file.
 2. **head:** It is used to read the first ten lines of file. (10 lines by default) The number of lines to be displayed may be specified in the head command.

Syntax:

\$head -n filename (-n means how many lines to be displayed)

Example:

\$head -20 student

Displays the first 20 lines of the file student.

3. **tail** : It is used to print last few numbers of lines (10 lines by default) of a certain file, then terminates.

Syntax:

\$tail {[+/-]n [lbc]} filename

Options	Meaning
---------	---------

+n	Displays all lines starting from nth line.
----	--

-n	Displays n lines from end of the file
----	---------------------------------------

l	Indicates Lines
---	-----------------

b	Indicates blocks
---	------------------

c	Indicates characters
---	----------------------

Example:

\$tail -20l student

Displays last 20 lines of student file.

\$tail +20l student

Displays all lines starting from 20th line of the student file.

4. **cut**: It is used to copy the specified columns to the standard output file. It is used to cut portion of a file. It takes filenames as command line arguments or input from standard input. It does not delete the selected portion of a file.

Syntax: \$cut [-f/-c] [n1,n2] filename

Where n1,n2 are the field/character number.

-f	Displays specified field. Also uses -d for field separator.
----	---

-c	Displays specified column character by character. Example 1:
----	--

\$cat student1

Ajay 1990 50

Vijay 1992 60

Sujit 1991 70

\$cut -f1,2 student1

Ajay 1990

Vijay 1992

Sujit 1991

In this output it will display 1st and 2nd fields values

Example 2:

```
$cat numbers 12345
```

```
23123
```

```
45656
```

```
76543
```

```
$cut -c4,5 student2
```

```
45
```

```
23
```

```
56
```

```
43
```

In this output it displays 4th and 5th character from each line.

5. **Paste** :It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output. When no file is specified, or put dash (“-“) instead of file name, paste reads from standard input and gives output as it is until a interrupt command [Ctrl-c] is given.

Syntax:

paste [OPTION]... [FILES]...

Create files for demonstration:

```
$cat stdname
```

```
Ajay
```

```
Vijay
```

```
Sujit
```

```
$cat birthyear
```

```
1980
```

```
2001
```

```
1996
```

```
$paste stdname birthyaer
```

```
Ajay  1980
```

```
Vijay  2001
```

```
Sujit  1996
```

6. **sort** : command is used to sort a file,

Syntax: sort [options] filename

arranging the records in a particular order. By default, the sort command sorts file assuming the

contents are ASCII. Using options in sort command, it can also be used to sort numerically.

When we have a mix file with both uppercase and lowercase letters then first the lower case letters would be sorted following with the upper case letters .

Option	Meaning
--------	---------

-o	write the output to a new file
-n	sort a file numerically
-r	Sort in Reverse Order

Example:

```
administrator@GFG19566-LAPTOP:~$ cat example.txt
apple
banana
cherry
date
```

```
administrator@GFG19566-LAPTOP:~$ sort -r example.txt
date
cherry
banana
apple
```

7. **uniq**: uniq command in Linux is a command-line utility that reports or filters out the repeated lines in a file.

Syntax: `uniq [OPTION]... [INPUT [OUTPUT]]`

Options	Meaning
---------	---------

-c	Precedes each output line with a count of the number of times each line appeared in the input file.
-d	Displays only the repeated lines.
-u	Displays only the unrepeated lines

Example :

\$cat kt.txt

I love music.

I love music.

I love music.

I love music of Kartik.

I love music of Kartik.

\$uniq kt.txt

I love music.

I love music of Kartik.

8. **tr (Unix):-** tr is a command in Unix-like operating systems. It is an abbreviation of translate or transliterate, indicating its operation of replacing or removing specific characters in its input data set.

Syntax : **\$ tr [OPTION] SET1 [SET2]**

Example:

```
bosko@bosko:~$ echo "Welcome" | tr e o
Wolcomo
bosko@bosko:~$
```

Here all **p,r,c,u** are converted into uppercase **P,R,C,U**.

```
sssit@JavaTpoint:~$ cat exm.txt | tr 'prcu' 'PRCU'
APPLE is Red.
Mango is yellow.
youR dReSS CoLoUR is Red.
Red CoLoUR suits on all.
```

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

3. Create a text file with ten lines of text in it.
4. Print first 5 lines of file (use head command)
5. Print last 4 lines of file (use tail command)

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. How many lines will be displayed with head command if number is not specified.
2. What are different types of filters used in Linux?
3. Give applications of paste Command.

(Space for answer)

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

XI. Exercise:

4. Create file with following contents

```
$ cat file2
```

Chhatrapati Shahu Maharaj

Dr.B.R.Ambedkar

Budhha

Dr.B.R.Ambedkar

Budhha

Dr.B.R.Ambedkar

Budhha

Write output of commands

- a) `$ uniq file2`
- b) `$ sort file2`
- c) `$ sort file2 | uniq`

(Space for answer)

XII. References/Suggestions for further reading: include websites/links

1. <https://www.thegeekstuff.com/2012/12/linux-tr-command>
2. <https://www.tecmint.com/wc-command-examples/>
3. <http://web.stanford.edu/class/cs273a/presentations.aut11/UnixTextProcessingPrimer.pdf>
4. <https://www.ibm.com/developerworks/aix/library/au-unixtext/index.html>

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....
4.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 13: Execute filters commands in Linux.

I. Practical Significance: A filter is a program that takes a flow of data from the standard input, processes or filters it and send the result to standard output.

II. Industry / Employer Expected Outcome:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning Outcomes (COs):

CO5 – Write programs using shell script.

IV. Laboratory Learning Outcome:

Execute commands grep, egrep and sed in Linux

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Syntax: `$grep <option> <pattern> <filename>`

Options	Meaning
---------	---------

- | | |
|----|--|
| -c | Displays the count of the number of occurrences of the pattern |
| -l | Displays the list of the filenames which contains a pattern. |
| -n | Displays line numbers along with the lines containing a pattern. |
| -v | Displays the lines which do not contain given pattern. |

\$cat OS

Unix

Linux

Ubuntu

\$grep -n Unix OS

It searches Unix pattern and displays line number where that pattern matched.

```
$ cat File1.txt
This is my first file.
Second line.
```

```
$ cat File4.txt
This is fourth file
Welcome to Linux
Hello World!
Good Bye!
```

Options:

\$ grep -c <pattern> <files>: This prints total number of lines that match the pattern in given files.

```
$ grep -c | File4.txt
2
```

\$ grep -n <pattern> <files>: This prints line that matches pattern along with its line number.

```
$ grep -n This File1.txt File4.txt
File1.txt:1:This is my first file.
File4.txt:1:This is fourth file
```

\$ grep -v <pattern> <files>: This inverts the sense of matching. It will show the lines which do not contain given pattern.

```
$ grep -v This File4.txt
Welcome to Linux
Hello world!
Good Bye!
```

2) **egrep:** The egrep command in Linux is a powerful tool used for searching and manipulating text based on patterns. It stands for 'Extended Global Regular Expressions Print', and as the name implies, it extends the capabilities of the original grep command by supporting additional regular expression syntax. It is commonly used for searching through text files and filtering output from other commands.

Syntax : egrep [options] 'pattern' [file...]

options: This is where you specify any flags or options that change how the egrep command behaves.

pattern: This is the regular expression that egrep will search for. It must be enclosed in quotes.

file: This is the file(s) in which egrep will search for the pattern. If no file is specified, egrep reads from the standard input.

Example:

```
anonny@ubuntu:~$ egrep page text.txt
man is the system's manual page. Each page argument given to
manual page associated with each of these arguments is then found
order (see DEFAULTS), and to show only the first page found, even
if page exists in several sections.
by the types of pages they contain.
Manual pages are normally stored in nroff(1) format under a
may also be preformatted cat pages to improve performance. See
This package supports manual pages in multiple languages,
```

3) **sed** :command in UNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening them, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.

SED command in unix supports regular expression which allows it perform complex pattern matching.

Syntax: sed OPTIONS [SCRIPT] [INPUTFILENAME]

Example: transliterate 'a-j' into '0-9':

```
$ echo hello world | sed 'y/abcdefghijklmnopqrstuvwxyz/0123456789/'
```

```
74llo worl3
```

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

6. Create a text file with ten lines of text in it.
7. Use grep , egrep and sed commands on above file.

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Give use of grep command.
2. Explain different options of grep command.
3. Compare grep and egrep command.

(Space for answer)

This image shows a full page of white paper with horizontal dashed lines, typical of primary school handwriting practice paper. The lines are evenly spaced and run across the entire width of the page. There are no margins, text, or other markings present.

XI. Exercise:

1.Create a text file with following contents

```
onworks@onworks-Standard-PC-i440FX-PiIX-996:-$ cat > ik.txt
iiiiiiiiiiiii
#abcbabcabc#
Interview
Kickstart
|
```

2) Replace the first instance of a pattern 'abc' with another pattern 'def' using sed command

(Space for answer)

[illegible]

XII. References/Suggestions for further reading: include websites/links

5. <https://www.thegeekstuff.com/2012/12/linux-tr-command>
6. <https://www.tecmint.com/wc-command-examples/>
7. <http://web.stanford.edu/class/cs273a/presentations.aut11/UnixTextProcessingPrimer.pdf>
8. <https://www.ibm.com/developerworks/aix/library/au-unixtext/index.html>

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 14: Execute shell scripts.**I. Practical Significance: shell scripts**

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning outcome(s)

CO5 - Write programs using shell script.

IV. Laboratory Learning outcome(s)

Read user input, exit and exit status commands, expr, and logical operators in shell scripts.

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions.
3. Follow ethical practices

VI. Relevant Theoretical Background (with diagrams if required)**1) Read User Input**

To read the Bash user input, we use the built-in Bash command called **read**. It takes input from the user and assigns it to the variable. It reads only a single line from the Bash shell. Below is the syntax for its implementation.

Syntax

```
read <variable_name>
```

2) exit and exit status commands

- The **exit** command terminates a script, just as in a C program. It can also return a value, which is available to the script's parent process.

- Every command returns an *exit status* (sometimes referred to as a *return status* or *exit code*). A successful command returns a 0, while an unsuccessful one returns a non-zero value that usually can be interpreted as an *error code*.

3) Expr command

The expr command in Unix evaluates a given expression and displays its corresponding output. It is used for:

- Basic operations like addition, subtraction, multiplication, division, and modulus on integers.
- Evaluating regular expressions, string operations like substring, length of strings etc.

Syntax:

\$expr expression

4) logical operators

They are also known as boolean operators. These are used to perform logical operations. They are of 3 types:

- **Logical AND (&&):** This is a binary operator, which returns true if both the operands are true otherwise returns false.
- **Logical OR (||):** This is a binary operator, which returns true if either of the operand is true or both the operands are true and return false if none of them is false.
- **Not Equal to (!):** This is a unary operator which returns true if the operand is false and returns false if the operand is true.

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Procedure: Teacher must assign a separate statements to students

- 1) Calculating the sum of two variables
- 2) Finding the remainder of two variables
- 3) Calculating the average of a list of numbers

IX. Result(s)

.....
.....
.....

References/Suggestions for further reading:

<https://www.computerhope.com/unix/uchmod.htm>

<https://www.geeksforgeeks.org/>

X. Practical related questions (Provide space for answers)

- 1) What is shell scripting?
- 2) What is the importance of shell scripting?

.....
.....
.....

Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
- 2.....
- 3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 15: . Execute the shell script by using the if statement.

- I. Practical Significance:** If is a statement that allows the programmer to make a decision in the program based on conditions he specified. If the condition is met, the program will execute certain lines of code otherwise, the program will execute other tasks the programmer specified.
- II. Industry / Employer Expected Outcome:**
 - 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
 - 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
 - 3) Ability to perform file operations and manipulate directories.
- III. Course Level Learning Outcomes (COs):**
CO5 –Write programs using shell script.
- IV. Laboratory Learning Outcome:**
Write the shell script by using the “if” statement
- V. Relevant Affective Domain related outcome(s)**
 - 1. Follow precautionary measures.
 - 2. Follow naming conventions
 - 3. Follow ethical practices.
- VI. Relevant Theoretical Background**

The if...else...fi statement is the control statement that allows Shell to execute statements in a controlled way and make the right choice.

Syntax;

if [expression] ; then

Statement(s) to be executed if expression is true

else

Statement(s) to be executed if expression is not true

fi

The Shell expression is evaluated in the above syntax. If the resulting value is true, given statement(s) are executed. If the expression is false, then no statement will be executed.

Relational operators are used along with if statement.

Relational operators used by if

Operator	Meaning
-eq	Equal to
-ne	Not equal
-gt	Greater than
-ge	Greater than or equal to
-lt	Less than
-le	Less than or equal to

General Syntax

1.Single Decision: -

Syntax:-

```
if [ condition ]; then
    ### series of code goes here
fi
e.g.
if [ "$X" -lt "0" ]; then
    echo "X is less than zero"
fi
```

2.Double Decision: -

Syntax:-

```
if [ condition ]; then
    Statements if the condition is satisfied
else
    Statements of code if the condition is not satisfied
fi
```

3.Multiple if condition: -

Syntax:-

```
if [ condition1 ]; then
    Statements for condition1
elif [ condition2 ]; then
    Statements for condition2
```

```
else
    Statements if the condition is not satisfied
fi
```

4.Double –bracket:-

Syntax:-

```
if ((condition)) then
    Statements goes here
fi
```

To type any program in Linux you need the compiler. vi editor is used for creating file in Linux.

Example :

Write a shell script to display welcome message with today's date in it.

In vi editor write a program code and save and quit with filename 'sample.sh'.

\$vi sample.sh

(Now press escape and i to change the input mode and type following program)

```
#sample.sh
#use of if-then statement
if whoami; then
    var1=`whoami`
    echo "Welcome $var1"
fi
if date; then
    echo "It displays todays date"
fi
echo "This is end of script"
```

To run the script

./sample.sh or \$ sh sample.sh

Save this program press escape+shift+:(colon) and type wq (:wq)

“Welcome User1”

Mon Feb 5 22:29:29 PST 2023

“It displays todays date”

“This is end of script”

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

1. Execute shell script by considering example like giving passing grades to students using if statement:

1. Single Decision.
- 2.Double Decision.
- 3.Multiple if statements.

Note: Attach the code at the end.

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

2. Write and execute script for nested if statements..
3. Write script for finding greatest number among given three number.

(Space for answer)

[illegible]

XI. Exercise:

Correct the following script and write its output.

1. if [! -r "\$1"] then
 echo "File \$1 is not readable – skipping."
fi
2. if ["\$X" -nt "/etc/passwd"]; Then
 echo "X is a file which is newer than /etc/passwd"
fi

(Space for answer)

XII. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>
4. <https://www.shellscript.sh/test.html>

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

1.....

2.....

3.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 16: . Execute the shell script by using the while statement.

I. Practical Significance: The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.

II. Industry / Employer Expected Outcome:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning Outcomes (COs):

CO5 –Write programs using shell script.

IV. Laboratory Learning Outcome:

Write the shell script by using the “while” statement

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI. Relevant Theoretical Background

A while loop is a control flow statement in Bash scripting that allows a certain block of code to be executed repeatedly as long as a specified condition is true. The loop provides a way to automate repetitive tasks and is a fundamental construct in scripting and programming.

Syntax:

```
while [condition]; do
..
..
..
done
```

Example:

```
a=1
```

```
while [ $a -le 5 ]; do
    echo "value of a=" $a
    a=`expr $a + 1`
done
```

1.The break statements: The break statement is used to terminate the execution of the entire loop, after completing the execution of all of the lines of code up to the break statement using break keyword.

2.The continue statement: The continue statement is similar to the break command, except that it causes the current iteration of the loop to exit, rather than the entire loop.

Example:

```
$cat fact.sh
echo "Enter a number"
read num
fact=1
while [ $num -gt 1 ];
do
    fact=$((fact * num))
    num=$((num - 1))
done
echo "factorial of a number is $fact"
```

Save this program press escape+shift+:(colon) and type wq (:wq)

To compile the program

\$/fact.sh or \$sh fact.sh

Enter the number : 5

Factorial of a given number is 120

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and open source operating System. (RedHat, Ubuntu etc.).	All

VIII. Program code

1. Write a shell script to display Fibonacci series for n numbers.
2. Write a shell script to display multiplication table of 5

Note: Attach the code at the end.

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

4. Write a shell script to accept five digit number and perform addition of all digits.
5. Write script for printing even numbers between 1 to 20.

(Space for answer)

[illegible][illegible]

XI. References/Suggestions for further reading: include websites/links

5. <https://www.tutorialspoint.com/unix>
6. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
7. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>
8. <https://www.shellscript.sh/test.html>

XII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
- 2.....
- 3.....
- 4.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	

Experiment No 17: . Execute the shell script by using the for statement.

I. Practical Significance: The for loops iterate through a set of values until the list is exhausted. In simple terms the for loop operates on lists of items. It repeats a set of commands for every item in a list.

II. Industry / Employer Expected Outcome:

- 1) To understand the basics of Linux operating system fundamentals and its open-source nature.
- 2) Basic Scripting Skills for automating tasks and creating custom shell scripts.
- 3) Ability to perform file operations and manipulate directories.

III. Course Level Learning Outcomes (COs):

CO5 –Write programs using shell script.

IV. Laboratory Learning Outcome:

Write the shell script by using the “for” statement

V. Relevant Affective Domain related outcome(s)

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI. Relevant Theoretical Background

Syntax of FOR loop: -

```
for var in item1 item2 ... itemN do
    command1 command2
    ....
    ...
    commandN
done
```

Example:

```
for var in 0 1 2 3 4 5 6 7 8 9 do
```

```
    echo $var  
done
```

Syntax of for loop using in and list of values is shown below.

```
for varname in list  
do  
    echo "statments"  
done
```

This for loop contains a number of variables in the list and will execute for each item in the list. For example, if there are 10 variables in the list, then loop will execute ten times and value will be stored in varname.

VII. Required Resources/apparatus/equipment with specifications

No.	Equipment Name with Broad Specifications	Relevant LLO Number
	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and opensource operating System. (RedHat, Ubuntu etc.).	

VIII. Program code

1. Execute shell script by considering example like printing following output by FOR loop

```
*****  
*****  
**  
*
```

Note: Attach the code at the end.

IX. Result(s)

X. Practical related questions (Provide space for answers)

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Execute shell script for printing table of given number using FOR loop
2. Write a shell script to print pattern

1			
2	2		
3	3	3	
4	4	4	4

(Space for answer)

This image shows a full page of blank handwriting practice paper. It features multiple sets of horizontal lines. Each set consists of three lines: a solid top line, a dashed middle line, and a solid bottom line. These sets are repeated down the entire page, providing a guide for letter height and placement. The paper is otherwise completely blank, with no text or markings.

XI. Exercise:

Give output of the following a.

```
#!/bin/sh
```

```
NUMBERS="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
```

(Space for answer)

XII. References/Suggestions for further reading: include websites/links

1. <https://www.tutorialspoint.com/unix>
2. <http://www.ee.surrey.ac.uk/Teaching/Unix/>
3. <https://www.cs.sfu.ca/~ggbaker/reference/unix/>
4. <https://www.cyberciti.biz/faq/bash-for-loop/>
5. https://bash.cyberciti.biz/guide/For_loop

XIII. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		30%
1.	Logic Formation	10%
2.	Debugging Ability	15%
3.	Follow ethical practice	5%
Product related (15 Marks)		70%
4.	Interactive GUI	20%
5.	Answer to Practical related questions	20%
6.	Expected Output	20%
7.	Timely Submission	10%
Total(25 Marks)		100%

List of student Team Members

- 1.....
2.....
3.....
4.....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	