MASTER OF COMPUTER APPLICATIONS

UNIX LAB

TEACHER'S LAB MANUAL

II YEAR I SEMESTER

Faculty In-Charge

Dr. M. Kalpana Devi
Sr. Asst. Professor, MCA Department

SREENIVASA INSTITUTE OF TECHNOLOGY
AND
MANAGEMENT STUDIES
(Autonomous)
(Approved by AICTE, New Delhi, Affiliated to JNTUA, Ananthapuramu, Accredited by NAAC, Bangalore)
Murukambattu, Chittoor – 517127

2018-19

INSTITUTE VISION AND MISSION

INSTITUTE VISION

To emerge as a Centre of Excellence for Learning and Research in the domains of engineering, computing and management.

INSTITUTE MISSION

- Provide congenial academic ambience with state-art of resources for learning and research.
- Ignite the students to acquire self-reliance in the latest technologies.
- Unleash and encourage the innate potential and creativity of students.
- Inculcate confidence to face and experience new challenges.
- Foster enterprising spirit among students.
- Work collaboratively with technical Institutes / Universities / Industries of National and International repute

DEPARTMENT VISION AND MISSION

DEPARTMENT VISION

To become the Centre of excellence for skilled software professionals in Computer Applications.

DEPARTMENT MISSION

- Provide congenial academic ambiance with necessary infrastructure and learning resources.
- Inculcate confidence to face and experience new challenge from industry and society
- Ignite the students to acquire self reliance in the State-of-the Art Technologies.
- Foster Enterprise spirit among students

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

Graduates of Computer Applications shall

PEO1: Have Professional competency through the application of knowledge gained from fundamental and advanced concepts of structural and functional components in software. **(Professional Competency)**

PEO2: Excel in one's career by critical thinking toward successful services and growth of the organization or as an entrepreneur or through higher studies. **(Successful Career Goals)**

PEO3: Enhance Knowledge by updating advanced technological concepts for facing the rapidly changing world and contribute to society through innovation and creativity. **(Continuing Education to Society)**

PROGRAM OUTCOMES (PO's)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

LAB COURSE SYLLABUS, LAB COURSE OUTCOMES, CO Vs PO

III MCA - I Semester	**L**	**T**	**P**	**C**
	0	**0**	**3**	**2**

16MCA217 **UNIX LAB**

PREREQUISITES: A course on "**Operating System**".

Course Educational Objectives:

CEO1 To Practice Basic Unix Commands for Files and Directories.

CEO2 To Practice Vi editor and to know about awk .

CEO3 To Explore Basic Shell Script Programs.

Syllabus:

Practice the Following commands in UNIX

1. Entering commands.
2. Common Commands for Files and Directories
3. Searching Files
4. More about Listing Files
5. Permission Commands
6. Commands for viewing Long Files, to Print Files
7. Editing with vi Editor
8. Finding Patterns in Files
9. Compressing and Packing Files
10. Counting Lines , words and File Size
11. Working with Columns and Fields
12. Sorting the Contents of Files
13. Comparing Files
14. Editing and Formatting Files
15. Working with Dates and Times
16. Performing Mathematical Calculations.
17. Standard input and Output (Redirection Commands)
18. awk Utilities.
19. Write a Shell Script that copies multiple files to a directory.
20. Write a Shell Script (small calculator) that adds, subtracts, multiplies and divides the given two integers. There are two division options: one returns the quotient and the other returns reminder. The Script requires 3 arguments : The operation to be used and

two integer numbers. The options are add(-a), subtract(-s), multiply(-m), quotient(-c) and reminder (-r).

21. Write a Shell Script that counts the number of lines and words present in a given file.
22. Write a Shell Script that displays the list of all files in the given directory.
23. Write a Shell Script to generate a Multiplication Table.
24. Write a Shell Script to reverse the rows and columns of a matrix.

Course Outcomes:

On successful completion of this course, students will be able to:

COURSE OUTCOMES		POs related to COs
CO1	**Demonstrate** knowledge on working in the Unix environment	PO1
CO2	**Analyze** the way in which programs and files are manipulated in the Unix environment	PO2
CO3	**Design** and **develop** algorithms and programs using various shell scripting .	PO3
CO4	**Conduct** investigation and test the networking commands in managing multiple users	PO4
CO5	Use appropriate design tools to **understand** the networking implementation in the Unix multiuser environment	PO5
CO6	**Follow** ethical principles in **design**ing and implementing multitasking and multiuser environment.	PO8
CO7	Do **experiment**s effectively as an individual and as a member in a group.	PO9
CO8	**Communicate** verbally and in written form, the understandings about the experiments.	PO10
CO9	Continue **updat**ing their skill related to Verilog HDL and FPGA implementation for various application during their life time	PO12

CO- PO MAPPING

Course	PO / CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
UNIX Lab	CO1	3	-	-	-	-	-	-	-	-	-	-	-
	CO2	-	3	-	-	-	-	-	-	-	-	-	-
	CO3	-	-	3	-	-	-	-	-	-	-	-	-
	CO4	-	-	-	3	-	-	-	-	-	-	-	-
	CO5	-	-	-	-	3	-	-	-	-	-		-
	CO6	-	-	-	-	-	-	-	3	-	-	-	-
	CO7	-	-	-	-	-	-	-	-	3	-	-	-
	CO8	-	-	-	-	-	-	-	-	-	3	-	-
	CO9	-	-	-	-	-	-	-	-	-	-	-	3

Course Outcome Attainment (R13/R16)

Day – To – Day Evaluation)	Fair	**Level 1**	If Student scored less than 80% of the total mark allotted.
	Good	**Level 2**	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	**Level 3**	If Student scored greater than 90% of the total mark allotted.
Term End Exam (TEE)	Fair	**Level 1**	If Student scored less than 80% of the total mark allotted.
	Good	**Level 2**	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	**Level 3**	If Student scored greater than 90% of the total mark allotted.

Course Outcome Attainment (R18)

Day – To – Day Evaluation	Fair	**Level 1**	If Student scored less than 80% of the total mark allotted.
	Good	**Level 2**	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	**Level 3**	If Student scored greater than 90% of the total mark allotted.
Internal Practical Exam	Fair	**Level 1**	If Student scored less than 80% of the total mark allotted.
	Good	**Level 2**	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	**Level 3**	If Student scored greater than 90% of the total mark allotted.
Term End Exam (TEE)	Fair	**Level 1**	If Student scored less than 80% of the total mark allotted.
	Good	**Level 2**	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	**Level 3**	If Student scored greater than 90% of the total mark allotted.

RUBRICS FOR UNIX LAB

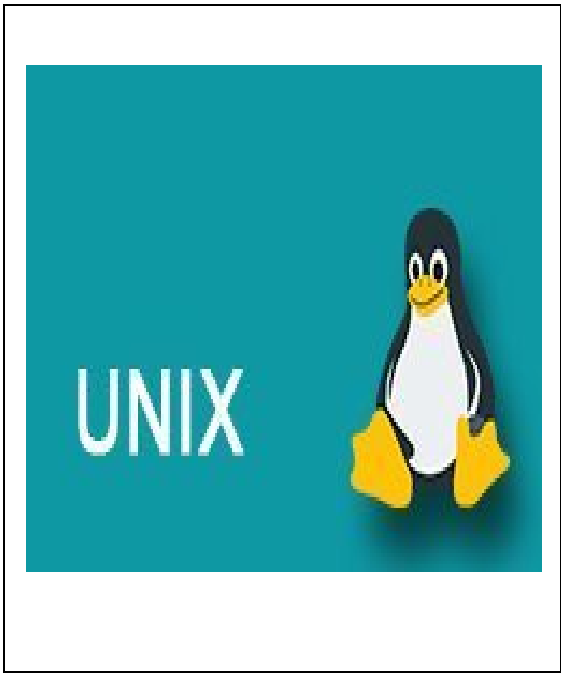
	Excellent(3)	Good(2)	Fair(1)
Conduct Experiments (CO1)	Student successfully completes the experiment, records the data, analyzes the experiment's main topics, and explains the experiment concisely and well.	Student successfully completes the experiment, records the data, and analyzes the experiment's main topics	Student successfully completes the experiment, records the data, and unable to analyzes.
Analysis and Synthesis (CO2)	Thorough analysis of program developed designed	Reasonable analysis of program developed	Improper analysis of program developed
Design (CO3)	Student understands what needs to be tested and designs an appropriate experiment, and explains the experiment concisely and well	Student understands what needs to be tested and designs an appropriate experiment.	Student understands what needs to be tested and does not design an appropriate experiment.
Complex Analysis & Conclusion (CO4)	Thorough comprehension through analysis/ synthesis	Reasonable comprehension through analysis/ synthesis	Improper comprehension through analysis/ synthesis
Use modern tools in executing the programs (CO5)	Student uses the tools to develop and execute the programs, and understands the limitations of the tool.	Student uses the tools correctly.	Student uses the tools correctly, unable to understand properly.
Report Writing (CO6)	Status report with clear and logical sequence of parameter using excellent language	Status report with logical sequence of parameter using understandable language	Status report not properly organized
Lab safety (CO7)	Student will demonstrate good understanding and follow lab safety	Student will demonstrate good understanding of lab safety	Students demonstrate a little knowledge of lab safety.
Ability to work in teams (CO8)	Performance on teams is excellent with clear evidence of equal distribution of tasks and effort	Performance on teams is good with equal distribution of tasks and effort	Performance on teams is acceptable with one or more members carrying a larger amount of the effort
Continuous learning (CO9)	Highly enthusiastic towards continuous learning	Interested in continuous learning	Inadequate interest in continuous learning

SREENIVASA INSTITUTE of TECHNOLOGY and MANAGEMENT STUDIES (autonomous)

UNIX LAB

LABORATORY MANUAL

II MCA- I SEMESTER Regulation: 16



FACULTY INCHARGE: DR. M. KALPANA DEVI
DESIGNATION : SR. ASST. PROFESSOR
DEPARTMENT: MCA

SITAMS	UNIX LABLABORATORY	Subject Code :16MCA217

INDEX

Sl. No.	Date	Name of the Experiment/Exercise	Page No.	Marks	Signature
1		Entering commands.			
2		Common Commands for Files and Directories			
3		Searching Files			
4		More about Listing Files			
5		Permission Commands			
6		Editing with vi Editor			
7		Finding Patterns in Files			
8		Compressing and Packing Files			
9		Counting Lines , words and File Size			
10		Working with Columns and Fields			
11		Sorting the Contents of Files			
12		Comparing Files			
13		Editing and Formatting Files			
14		Working with Dates and Times			
15		awk Utilities.			
16		Write a Shell Script that copies multiple files to a directory.			
17		Write a Shell Script (small calculator) that adds, subtracts, multiplies and divides the given two integers. There are two division options: one returns the quotient and the other returns reminder. The Script requires 3 arguments : The operation to be used and two integer numbers. The options are add(-a), subtract(-s), multiply(-m), quotient(-c) and reminder (-r).			
18		Write a Shell Script that counts the number of lines and words present in a given file.			
19		Write a Shell Script that displays the list of all files in the given directory.			
20		Write a Shell Script to generate a Multiplication Table.			
21		Performing Mathematical Calculations			

Signature of the faculty in-charge with date

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
MCA Department

Name:		**RollNo:**			**Year&Sem :**			**AY:**
SNO.	**Experiment Name**	**Knowledge Gained**	**Analysis, Design and use of Modern Tool/Technique**	**Ability of doing experiment and following of ethical principles**	**Result &Conclusion**	**VIVA VOCE (Communication, Long Learning)**	**Total**	**Signature of the Faculty**
		10	**10**	**5**	**5**	**10**	**40**	
1	Entering commands.							
2	Common Commands for Files and Directories							
3	Searching Files							
4	More about Listing Files							
5	Permission Commands							
6	Editing with vi Editor							
7	Finding Patterns in Files							
8	Compressing and Packing Files							
9	Counting Lines , words and File Size							
10	Working with Columns and Fields							
11	Sorting the Contents of Files							
12	Comparing Files							
13	Editing and Formatting Files							
14	Working with Dates and Times							
15	awk Utilities.							
16	Write a Shell Script that copies multiple files to a directory.							
17	Write a Shell Script (small calculator) that adds, subtracts, multiplies and divides the given two integers. There are two division options: one							

	returns the quotient and the other returns reminder. The Script requires 3 arguments : The operation to be used and two integer numbers. The options are add(-a), subtract(-s), multiply(-m), quotient(-c) and reminder (-r).							
18	Write a Shell Script that counts the number of lines and words present in a given file.							
19	Write a Shell Script that displays the list of all files in the given directory.							
20	Write a Shell Script to generate a Multiplication Table.							
21	Performing Mathematical Calculations							
Total Attainment								
% of Attainment								
Level								

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
MCA Department

S.No	Experiment Name	CO1	CO2	CO3	CO4	CO5	CO6	CO7	CO8	CO9
		Knowledge	Analysis	Design	Complex Analysis & Conclusion	Use of modern tools	Communication ability	Ethics	Individual / Team work	Life Long Learning
1	Entering commands.									
2	Common Commands for Files and Directories									
3	Searching Files									
4	More about Listing Files									
5	Permission Commands									
6	Editing with vi Editor									
7	Finding Patterns in Files									
8	Compressing and Packing Files									
9	Counting Lines , words and File Size									
10	Working with Columns and Fields									
11	Sorting the Contents of Files									
12	Comparing Files									
13	Editing and Formatting Files									
14	Working with Dates and Times									
15	awk Utilities.									
16	Write a Shell Script that copies multiple files to a directory.									
17	Write a Shell Script (small calculator) that adds, subtracts, multiplies and divides the given two integers. There are two division options: one returns									

	the quotient and the other returns reminder. The Script requires 3 arguments : The operation to be used and two integer numbers. The options are add(-a), subtract(-s), multiply(-m), quotient(-c) and reminder (-r).									
18	Write a Shell Script that counts the number of lines and words present in a given file.									
19	Write a Shell Script that displays the list of all files in the given directory.									
20	Write a Shell Script to generate a Multiplication Table.									
21	Performing Mathematical Calculations									
Average of Day– to – Day evaluation (C1)										

Aim:

To Perform Entering Commands

Commands:

Passwd : This command is used to his or her password

Syntax: $ passwd

Cal : This command display a calendar for any month or year.

Syntax: $Cal 2018

Who: This command is used to know the login details of all the current users.

Syntax: $who

Finger : This command provides complete information about other users on the system

Syntax: $finger

Write: This command is used to send short message to another user

Syntax: $write username

Talk: This command is an enhanced communication program

Syntax: $talk username

Mesg: This command is used to accept or refuse messages sed via write or talk command

Syntax: $mesg username

Man: This command is used to display more information about other users

Syntax: $man

Date: This command is used to display the current day date

Syntax: $date

Clear: This command is used toclear the full screen

Syntax: $clear

Exit: This command is used to exit to the terminal

Syntax: $exit

1. ENTERING COMMANDS

```
[mca@linux ~]$  finger
        Login          Name      Try Idle Login Time   Office     Office Phone
        mca26           pts/2      Nov 17   12:00                  (10.1.5.46)
        mca30           pts/1      Nov 17   11:44                  (10.1.5.48)

[mca@linux ~]$ who
        mca30    pts/1       2017-11-10 11:44 (10.1.5.48)
        mca26    pts/2       2017-11-10 12:00 (10.1.5.46)
```

```
        mca45    pts/3        2017-11-10 12:02 (10.1.5.49)

[mca@linux ~]$ passwd
        Changing password for user mca26.
        Changing password for mca26
        (current) UNIX password: *******
        New UNIX password:  *******
        Retype new UNIX password:  *******
        passwd: all authentication tokens updated successfully.
[mca@linux ~]$ cal
        November 2017
        Su Mo Tu We Th Fr Sa
        1  2  3  4
        5  6  7  8  9 10 11
        12 13 14 15 16 17 18
        19 20 21 22 23 24 25
        26 27 28 29 30

[mca@linux ~]$ cal 6 2017
          June 2017
        Su Mo Tu We Th Fr Sa
        1  2  3
        4  5  6  7  8  9 10
        11 12 13 14 15 16 17
        18 19 20 21 22 23 24
        25 26 27 28 29 30
[mca@linux ~]$    write mca43
Hai , all the best to every one  ……,
CTRL+D
[mca@linux ~]$ talk mca43
INBOX                                          SENT MGS
Mca: Hai , all the best to every one                    Mca44:thank you,
[mca@linux ~]$         mesg  y
[mca@linux ~]$         mesg  n
[mca@linux ~]$ man cat
        CAT(1)   User Commands   CAT(1)
        NAME
        cat - concatenate files and print on the standard output
        SYNOPSIS    cat [OPTION] [FILE]...
        DESCRIPTION Concatenate FILE(s), or standard input, to standard output.
        -A, --show-all
        equivalent to –vET
[mca@linux ~]$  date
        Thu Nov 10 12:20:34 IST 2017

[mca@linux ~]$  clear
        clear the full screen.

[mca@linux ~]$  Exit
        Exit to the terminal.
```

Aim:

To perform files and directories commands

Commands:

Ls: This command is used to see all the files in the directory.

Syntax:

- ls -a: list all files including hidden files. These are files that start with ".".
- ls -A: list all files including hidden files except for "." and ".." – these refer to the entries for the current directory, and for the parent directory.
- ls -R: list all files recursively, descending down the directory tree from the given path.
- ls -l: list the files in long format i.e. with an index number, owner name, group name, size, and permissions.
- ls – o: list the files in long format but without the group name.
- ls -g: list the files in long format but without the owner name.
- ls -i: list the files along with their index number.
- ls -s: list the files along with their size.
- ls -t: sort the list by time of modification, with the newest at the top.
- ls -S: sort the list by size, with the largest at the top.
- ls -r: reverse the sorting order.

Cat: This command is used to create a new file in current directory.

$cat> filename // to create new file

$cat filename // to view content of file

$cat> file1 file2 file3 > destination // combines one or more files into another file

$cat> file1>>file2 //add or appends the content of file2

$cat> file@ > destination // displays list of files with similar names.

Pwd: This command is used to view current directory

$pwd -L: Prints the symbolic path.
$pwd -P: Prints the actual path.

Cd: This command is used to move between directories.

$cd // to move between directories

$cd filename // to change from one directory to another

$cd .. // used to come out of the directory

$cd ~ // move to previous directory

Mv: This command is used to move or remove files.

$mv <source file> <destination file> // move file to new file.

$mv <file1> <file2> <file3>……. <destination file> //move all files into destination file.

Cp: This command is used to copy a file from directory to another.

$cp <source file > <destination file> // copy file from source file to destination

$cp -r <directory name1> <directory number> // copies file from one directory to another directory.

Ln: This command is used to create link between files which enables a single file at two or more locations in the directory system.

$ln /home /dkrout/project project // it creates a hard link for a file in /home/drkout.

$ls -I // to seeif two files are hard linked to each other.

$ls -s /home/dkrout/mca1 mca2 // it will create symbolic link called mca2

Rm : this command removes the named files from the file system

$rm <filename> //used to remove a specified file

$rm -r <directoryname> //used to remove directory with file

$rm -I <filename> //used to remove file to move safely.

Mkdir: This command is used to make a directory.

$mkdir <directory name>

$rmdir<directory name> // to remove directory with all the files.

$rmdir<directory name> // to remove directory with all the files with its content..

2. COMMON COMMANDS FOR FILES AND DIRECTORIES.

(i)ls COMMAND:

```
[mca@linux sitams] $ ls
        regfile1   regfile2   regfile3 dir1 dir2   sym.lnk

[mca@linux sitams] $ ls -F
        regfile1 regfile2 regfile3 dir1/ dir2/ sym.lnk@

[mca@linux sitams] $ ls -R
        .
        ..
        dir1  dir2  regfile1  regfile2  regfile3 sym.lnk
        ./dir1:
        dir1file
```

(ii)cat COMMAND:

```
[mca@linux sitams] $   cat > file1
        Unix is an operating system
        .
[mca@linux sitams] $   cat file2
```

cat command is to create a file in the current directory

[mca@linux sitams] $ cat file2 >> file1
[mca@linux sitams] $ cat file1
Unix is an operating system.
cat command is to create a file in the current directory

[mca@linux sitams] $ file *
regfile1: ASCII text
dir2: empty
dir1: directory
regfile2: ASCII text
regfile: ASCII text

(iii) MOVING AROUND DIRECTORIES:

[mca@linux ~] $ cd sitams // to move to sitams from current directory
[mca@linux sitams] $ cd ~ //to move to home directory
[mca@linux ~] $ pwd //to view the Present Working Directory
/home/mca26
[mca@linux ~] $ cd .. [enter]
Move to previous directory.
[mca@linux ~]$ cd home/sitams/mca/2year [enter]
//to move to more directories at same time.

(iv) MOVING AND COPYING FILES:

[mca@linux sitams] $ cat file1
Unix is an operating system.
[mca@linux sitams] $ mv file1 file2 //moves the contents from file1 to file2
// content does not exists in file1
[mca@linux sitams] $ cat file2
Unix is an operating system.
[mca@linux sitams] $ cat file1
cat: file1:No such file or directory.
[mca@linux sitams] $ cp file1 file2 // the contents of file1 is copied to file2
//content exists in both file1 andfile2
[mca@linux sitams]$ cat file1
Unix is an operating system.
[mca@linux sitams]$ cat file2
Unix is an operating system.
[mca@linux sitams]$ rm file3 //the file3 is removed from current directory
rm: remove regular file ‘file3’?y
[mca@linux sitams]$ cat file3

```
        file3: work: No such file or directory
[mca@linux sitams]$ mkdir dir3                    //makes a new directory called dir3
[mca@linux sitams]$ cd dir3                    // move to dir3 directory
[mca@linux sitams dir3]$
[mca@linux sitams]$ rmdir dir3                    //removes the directory dir3
[mca@linux sitams] $   cd dir3
        -bash: cd: dir3: No such file or directory
```

Aim:

To search the contents in the one or more directories.

Searching Files:

- Using find.
- Running find in background.
- Other search criteria.

Using find:

'find' command used to search the contents in one or more directories including all sub directories.

Syntax:

1. $find /_name new. Data-print //search starts from root directory and display the files that matches filename.
2. $find/temp/project-name new. Data-print //searches in its current and its subdirectories.
3. $find.-name "*data"-print //searches from all files in current directory that ends with data.

Running find in background:

To achieve multitasking feature of Unix system we use this command.

Syntax:

$find /-name new. Data-print > found & //searches the file from room directory and results are send to found file.

Other search criteria:

Search criteria can be specified in the find command.

Syntax:

$find.-name "music" –u sue-m times +7-print //To search four a file called music belonging to user 'sue' that was modified more than a week ago.

3.SEARCHING COMMANDS.

```
[mca@linux sitams]$pwd                    //it displays present work directory.//
                  /home/unix

[mca@linux sitams]$   find . -name "mca*"
                                ./mca4
                                ./mca1
```

./mca3
./mca2

[mca@linux sitams]$ find . -name "mca*"> pgcourse
[mca@linux sitams]$ cat pgcourse
./mca4
./mca1
./mca3
./mca2

[mca@linux sitams]$ find .name “music” –u &ue –ontime +7 –print

-ontime //option is used to specify the number of days it hs been since the file was modified

-u //option is used to search the files to a particular..

Aim:

To perform listing files in various types.

Commands:

- Listing hidden files.
- Controlling the way ls display file names.
- Showing non printing character.
- Sorting listings.
- Combining option to ls and the long form to ls.

Listing hidden files:

Due to some reasons we have to hide the files. We can view the hidden files using "ls with –a".

Syntax:

1. $ls-a // list files including hidden files.
2. $ls // To list current directories.

Controlling the way ls display file names:

By default ls command displays files in multiple columns, sorted down the column as

Syntax:

$ls-x // To display file names horizontally in as many lines as necessary.

Showing non printing character:

For suppose, we have missed a single character while making a file, such missed character is said to be non printing character.

Syntax:

$ls-b // replaces a non printing character with its octal code.

Sorting listings:

Several options enable to control the order in which ls sorts its output.

Syntax:

1. $ls-t //most recently changed files are listed first.
2. $ls-r // lists files in reverse alphabetical order.

4.MORE ABOUT LISTING FILES.

ls options file|dir

options can be

Option	description

ls -a	list all files including hidden file starting with '.'
ls –color	colored list [=always/never/auto]
ls –d	list directories - with ' */'
ls –F	add one char of */=>@| to entries
ls –i	list file's inode index number
ls -l	list with long format - show permissions
ls -la	list long format including hidden files
ls -lh	list long format with readable file size
ls -ls	list with long format with file size
ls -r	list in reverse order
ls -R	list recursively directory tree
ls -s	list file size
ls -S	sort by file size
ls -t	sort by time & date
ls –X	sort by extension name

```
[mca@linux sitams]$   ls –a                          //includes hidden files also

      .   ..   regfile1   regfile2   regfile3  dir1  dir2   sym.lnk

[mca@linux sitams]$ ls –x                            //lists files row wise

      abc     biggest         cal
      dir1    dir2            dir3
      hello   sample1         sample2

[mca@linux sitams]$ ls –l                            //lists files in long form

      total 20
      -rw-rw-r-- 1    mca mca   56   Jun 16 14:16    regfile1
      -rw-rw-r-- 1    mca mca   29   Jun 16 14:22   regfile2
      drwxrwxr-x 2  mca mca   4096 Jun 16 14:07 dir1
      -rw-rw-r--1     mca mca    0    Jun 16 14:35   regfile3
```

```
[mca@linux sitams]$   ls –F                    // Lists files with marks that indicates the type of
file
        regfile1  regfile2   regfile3   dir1/   dir2/   sym.lnk@   mulline* // @ for symbolic link, /
        for
                                                                    directory*for executable file

[mca@linux sitams]$   ls –R            // Lists the contents of subdirectories also
        regfile1  regfile2   regfile3   dir1  dir2   sym.lnk   mulline
        dir1
                file1
                file2
        dir2
                file3
                file4

[mca@linux sitams]$   ls –S
        0 Regfile3  29 regfile2  56 regfile1  4096 dir1
```

Aim:

To perform the permission commands.

Permission commands:

Chmod command:

This command is used to alter a files permissions.

Syntax:

1. $chmod ugo+rwx quotations // To add permissions.

   $ls –l quotations.

2. $chmod go-rx quotations // To remove permissions.

   $ls –l quotations.

Setting permissions:

- Absolute permissions.
- Relative permissions.
- Umask.

Absolute permissions:

Absolute permissions are set by using a numeric code to specify them. Code represents a files permission by 3 digits.

Syntax:

$chmod 700 quotations // To set absolute permission.

$ls –l quotation.

Relative permission:

Relative permission are set using the ugo +/- rwx notation.

Syntax:

1. $chmod go –rwx * //to remove read, write and execute permissions except hidden files in current directory.
2. $chmod –R u+r email // changes to all of the files and subdirectories in directory.

Umask:

Umask is also used to alter file permissions. It uses a numeric code for representing absolute permissions.

Syntax:

1. $umask 777 // means read, write and execute permissions for user, group and others.
2. $umask 022 //all new files in this session will be given permissions.

5.CHANGING PERMISSION COMMANDS:

chmod is used to change the permissions of files or directories. Permission setting can be done in two ways

1) Relative Permission Setting
2) Absolute Permission Setting

1) Relative Permission Setting

General Form of Relative Permission Setting using chmod

Chmod *options permissions* filename

Following are the symbolic representation of *different options*:

- u is for user,
- g is for group,
- and o is for others.

Following are the symbolic representation of *different permissions*:

- r is for read permission,
- w is for write permission,
- x is for execute permission

S.No.	Chmod operator & Description
1	+ Adds the designated permission(s) to a file or directory.
2	- Removes the designated permission(s) from a file or directory.
3	= Sets the designated permission(s).

```
[mca@linux sitams] $$ls -l testfile
        -rwxrwxrwx  1 amrood   users 1024  Nov 12 00:10  testfile
[mca@linux sitams]$$chmod u-x testfile
[mca@linux sitams]$$ls -l testfile
        -rw-rwxrwx  1 amrood   users 1024  Nov 12 00:10  testfile
[mca@linux sitams] $$chmod g = rx testfile
[mca@linux sitams]$$ls -l testfile
        -rw-r-xrwx  1 amrood   users 1024  Nov 12 00:10  testfile
```

Here's how you can combine these commands on a single line −

```
[mca@linux sitams]$$chmod o+wx,u-x,g = rx testfile
[mca@linux sitams]$$ls -l testfile
        -rw-r-xrwx  1 amrood   users 1024  Nov 12 00:10  testfile
```

2) Obosulte Permission Setting

General Form of obsolute Permission Setting using chmod

Chmod 3digitnumber filename

Each digit is a combination of the numbers 4, 2, 1, and 0:

- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute", and
- 0 stands for "no permission."

So 7 is the combination of permissions 4+2+1 (read, write, and execute), 5 is 4+0+1(read, no write, and execute), and 4 is 4+0+0 (read, no write, and no execute).

ca@linux sitams]$ chmod 777 file1 // obsolute Permission Setting

-rwxrwxrwx 1 mca mca 28 Jun 16 14:37 file1

[mca@linux sitams]$ chmod 744 file2 // all permission to user and only read permission to group and others

-rwxr--r-- 1 mca mca 28 Jun 16 14:37 file1

Aim:

To edit a file using vi editor.

Procedure:

Step1: open a file for editing by typing

[mca@linux~] $vi sample

This is unix lab

The file is to illustrate the vi editor

Through which file can be edited.

Add lines

.....

This lines is added in the end of the file.

Step2: press 'esc:wq!' to save the file content and quit the editor.

Commands to edit the files:

Pointer moving commands:

K //moves the cursor up one line.

J // moves the cursor down one line.

H // moves the cursor to the left one character position.

L // moves the cursor to the right one character position

Editing commands:

i // inserts text before the current cursor location.

I // inserts text at the beginning of the current line.

a // inserts text after the current cursor location.

A // inserts text at the end of the current line.

o // creates a new line for text entry below the cursor location.

O // creates a new line for text entry above the cursor location.

6.EDITING WITH VI EDITOR:

[mca@linux ~]$ cat > sample

This is unix lab

This file is to illustrate the vi editor

Through which file can be edited

[mca@linux ~]$ cat sample

This is unix lab

```
        This file is to illustrate the vi editor

        Through which file can be edited

[mca@linux ~]$ vi sample

        This is unix lab

        This file is to illustrate the vi editor

        Through which file can be edited

        Add lines

        ********

        This line is added in the end of the file

[mca@linux ~]$ cat sample

        This is unix lab

        This file is to illustrate the vi editor

        Through which file can be edited

        Add lines

        ********

        This line is added in the end of the file
```

SITAMS	**EXPERIMENT: 7** **FINDING PATTERNS IN FILES**	**Page. No.**

Aim:-

To implement finding patterns in files.

Commands:-

1. The grep Command
2. The fgrep Command
3. The egrep Command

1.The grep Command:-

This grep command displays all the lines from the files which contains specified pattern.

Syntax:-

$ grep 'searchword/pattern' filename.

i. $grep–i // ignores the case i.e., matches either upper or lowercase.
ii. $grep–v // displays lines or files that doesn't contains search pattern.
iii. $grep –n // prints the matched line and its line numbers.
iv. $grep –l // prints only the name of file which matching lines.
v. $grep –c // prints only the count of matching lines.

2. The fgrep Command:-

This command searches for multiple targets and doesnot allow regular expressions.

Syntax:-

$fgrep "searchword1
searchword2
searchword3"filename.

7.FINDING PATTERNS IN FILES:

(i) grep COMMAND:

[mca@linux ~]$ cat > sample
This is unix lab
The commands searches pattern in a file
We are doing commands
[mca@linux ~]$ cat sample
This is unix lab
The commands searches pattern in a file
We are doing commands
[mca@linux ~]$ grep commands sample
The commands searches pattern in a file
We are doing commands
[mca@linux ~]$ grep "unix" sample *//displays line that contains "unix" pattern*
This is unix lab
[mca@linux ~]$ grep lab * *//displays files and corresponding lines where the pattern "lab"exists*

Sample:This is unix lab

[mca@linux ~]$ grep -i unix sample *// -i to ignore the case*

This is unix lab

[mca@linux ~]$ grep -v unix sample *// to display lines that does not contain "unix" pattern*

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ grep -n unix sample// *displays matched lines with line numbers*

1:This is unix lab

[mca@linux ~]$ grep -l unix sample *//displays the filenames which contains the pattern unix*

Sample

Demo

[mca@linux ~]$ grep -c unix sample// *displays number of lines that contains the pattern "unix" in sample file*

(ii) fgrep COMMAND:

[mca@linux ~]$ fgrep "unix

> pattern

> file" sample

This is unix lab

The commands searches pattern in a file

(iii) egrep COMMAND:

[mca@linux ~]$egrep "unix|file|commands" sample

This is unix lab

The commands searches pattern in a file

We are doing commands

SITAMS	**EXPERIMENT: 8** **COMPRESSING AND PACKING FILES**	**Page. No.**

Aim:-

To implement compressing and packing files.

Commands:-

1. The pack command

2 .The compress command

3. The gzip command

1. The pack command:-

- ❖ The pack command replaces a file with compressed version.
- ❖ The compressed file has '.z' extension appended to the filename, to indicate how it was compressed.

Syntax:-

$ pack filename

To uncompress the compressed file, use "unpack" command.

Syntax:-

$unpack filename

2. The compress command:-

- ❖ The "compress" command works much the same as 'pack' command.
- ❖ It adds '-z' [upper case] at the end of compressed file.

Syntax:-$compress filename

- ❖ It is also possible to uncompress the file using "uncompress" command.
  Syntax:-$uncompress filename

3. The gzip command:-

- ❖ The gzip commands also replace a file with a compressed version.
- ❖ A compressed file with gzip has '-gz' extension.

Syntax:-$gzip filename

- ❖ To uncompress the file using gzip, use either 'gzip –d' or 'gunzip'.

Syntax:-$gunzip filename

8.Compressing and packing Files:

```
[mca@linux ~]$ pack research_data
        pack:research_data 45.4% compression
[mca@linux ~]$ ls research*
        research_data.z
[mca@linux ~]$ unpack research_data
        unpack:research_data : unpacked
[mca@linux ~]$ ls research*
        research_data
```

```
[mca@linux ~]$ compress research_data
[mca@linux ~]$ ls research*
                research_data.Z
[mca@linux ~]$ uncompress research_data
[mca@linux ~]$ ls research*
                research_data
[mca@linux ~]$ ls research*
                research_data
[mca@linux ~]$ gzip –v research_data
                research_data :81.3% -- replaced with research_date.gz
[mca@linux ~]$ ls research*
                research_data.gz
[mca@linux ~]$ gunzip –v *.gz
                research_data : 81.3% -- replaced with research_data
[mca@linux ~]$ ls research*
                research_data
```

Aim:-

To implement counting lines, words and filesize in the file.

Commands:-

1 The wc command

2 The nl command

1 The wc command:-

The word count (wc) command prints the number of bytes, lines or words in a file.

i. $wc –c // size of file in bytes.
ii. $wc–w // number of words in files.
iii. $wc –l // number of lines in file that contains.
iv. $wc –L // lengthof the longest line.

2 The nl command:-

The 'nl' command is used to number each line in a file.

Syntax:-

$ nl filename>numbered

9.COUNTING WORDS AND LINES IN FILES:

wc-to perform word count

wc OPTION FILE

options can be

–l prints the number of lines in file
–w prints the number of words in the file
–c count the file size in bytes
–m count the number of charecters in the file
–L print only length of the longest line in the file.

wc and nl commands

[mca@linux ~]$ cat > sample

This is unix lab

The commands searches pattern in a file

We are doing commands

[mca@linux ~]$ cat sample

This is unix lab

The commands searches pattern in a file

We are doing commands

```
[mca@linux ~]$ wc -c sample
        79 sample
[mca@linux ~]$ wc -w sample
        15 sample
[mca@linux ~]$ wc -L sample
        39 sample
[mca@linux ~]$ grep commands sample | wc -l
        2
[mca@linux ~]$ nl sample
         1  This is unix lab
         2  The commands searches pattern in a file
         3  We are doing commands
```

Aim:-

To work with columns and fields.

Commands:-

UNIX system indicates a no. of tools designed to work with files organized in columns.

1. Cut command:-

- Cut command is used to exact specific fields from file.
- Specify –f option and field with field number to select.

Syntax:-

$cut –f2 filename//extracts second field from the file.

$cut –f1,3 filename//extracts 1 and 3 fields from the file.

$cut –f1-3 filename//extracts from 1st field to 3rd field.

$cut –d//extracts and displays using specified delimeter.

$cut –c3// filename//cut to identify fields in terms of character positions.

2. Column command:-

'colrm' is used to remove one or more columns from a file or set of files.

Syntax:-

$colrm 2 filename

3. The paste command:-

'paste' command joins files together line by line.

Syntax:-

$paste file1 file2 > file3

The above command joins file1 and file2 into file3.

$paste –d // used to specify along with the default delimeter.

4 The Join command:-

'Join' command joins together two existing files in the basis of a key field that contain entries common to both of them.

Syntax:-

$join file1 file2

The above command joins 2 files.

10.WORKING WITH COLUMNS AND FIELDS:

[mca@linux ~]$ cat > phoneno

dan	dnidz	1234
robin	rpele	5678
ben	bsquared	9876

```
[mca@linux ~]$ cut -f1,3- phoneno > new1

[mca@linux ~]$ cat new1

        dan     dnidz           1234
        robin   rpele           5678
        ben     bsquared        9876

[mca@linux ~]$ cat > states

        alabama
        alaska
        arizona
        arkansas
        California

[mca@linux ~]$ cat >state_abbr

        al
        ak
        az
        ar
        ca

[mca@linux ~]$ paste states state_abbr >states.imp

[mca@linux ~]$ cat states.imp

        Alabama     al
        alaska       ak
        arizona       az
        arkansas     ar
        california   ca
[mca@linux ~]$ cat > merch

        63A457          mans gold watch
        73B312          garnet ring
        82B119          sapphire pendant
[mca@linux ~]$ cat > costs

        63A457          125.50
        73B312          255.00
        82B119          534.75
[mca@linux ~]$ join merch costs

        63A457          mans gold watch         125.50
        73B312          garnet ring             255.00
        82B119          sapphire pendant        534.75




[mca@linux ~]$ cat columns

        smith   ctr             9898983999      ap
        john    tpt             9898291899      ap
        ram     ctr             9839898989      ap
        sita    chennai         9897878788      TN
        situ    Bangalore       9385935893      KN
```

```
[mca@linux ~]$ cut -f3 columns

        9898983999
        9898291899
        9839898989
        9897878788

[mca@linux ~]$ cut -f1,2 columns

        smith   ctr
        john    tpt
        ram     ctr
        sita    chennai
        situ    Bangalore

[mca@linux ~]$ cut -f1 columns|sort

        john
        ram
        sita
        situ

[mca@linux ~]$ cut -f2 columns

        ctr
        tpt
        ctr
        chennai
        Bangalore

[mca@linux ~]$ cut -f2 columns|sort|uniq –u

        Bangalore
        chennai
        tpt

[mca@linux ~]$ cut -f2 columns|sort|uniq –d
        Ctr
```

Aim:-

To implement sorting on the file contents.

Commands:-

1. The sort command:-

- ‘Sort’ command order(or) reorders the lines of a file.
- ‘Sort’ is also used to combine the contents of several files into a single sorted file.

Syntax:-

$sort –o// sort to save the results of a file.

$sort –f// Ignores uppercase &lowercase.

$sort –n// sort by numeric values, in ascending order.

$sort –r// reverse order of output.

$sort –u// eliminates duplicate lines in output.

2. The uniq command:-

This ‘uniq’ command filters or remove repeated lines from files.

Syntax:-

$ unique filename

11.SORTING CONTENTS OF THE FILE:

[mca@linux ~]$ cat > names
lincroft
summit
holmden
Middletown

[mca@linux ~]$ cat > names
Lincroft
Summit
holmden
Middletown

[mca@linux ~]$ sort names
holmden
Lincroft
middletown
Summit

[mca@linux ~]$ sort -rn names
Summit
middletown
Lincroft
Holmden

Aim:-

To implement comparing on files

Commands:-

The cmp command:-

- 'cmp' is the simplest way of the file comparison tool.
- It tells whether two files differ and if they do it reports the position in the file where the first difference occurs.

Syntax:- $ cmp file1 file2

'cmp' does not print anything if there are no difference in the files.

The comm command:-

The compare two sarted files and shows lines that are the same or different.

Syntax:-

$ comm file1 file2.

- Comm prints its output in 3 colums
- Lines unique to the first file
- Lines unique to the second file
- Lines found in both

$ comm-23 file1 file2//displays only the lines that are unique to the first file

The diff command:-

Compares two files,line by line and print out differences.

Syntax:-

$diff file1 file2

c//change(c) means there is change in the line

a//append indicates that file is added with 3lines.

12.COMPARING FILES:

[mca@linux ~]$ cat > notes

Nate,
heres the first draft of the plan
i think it needs more work

[mca@linux ~]$ cat >notes.more

Nate,
heres the first draft of the new plan
i think it needs more work
let me know what you think

[mca@linux ~]$cmp notes notes.more

notes notes.more differ: byte 36, line 2

[mca@linux ~]$ cat > cities1

newyork
palo alto
san fransico
seattle

[mca@linux ~]$ cat > citites2

palo alto
san fransico
santamonica
seattle

[mca@linux ~]$comm cities1 cities2

newyork
palo alto
 san fransico
santamonica
seattle

[mca@linux ~]$ diff notes notes.more

2c2
<heres the first draft of the plan

>heres the first draft of the new plan
3a4
> let me know what you think

[mca@linux ~]$ diff notes notes.more> difference

[mca@linux ~]$ cat difference

2c2
<heres the first draft of the plan

>heres the first draft of the new plan
3a4
> let me know what you think

[mca@linux ~]$ patch notes difference

patching file notes

Aim:-

To implement editing and formatting files.

Commands:-

There are many ways to edit and format files in the unix system.

The pr command:-

'pr' command is used to add a header to every page of a file

Syntax:-$pr headername

$pr |lp//adds header to the file when they are printed

$pr-d-n file1|lp//prints the file1 with double spaced and with line numbers

$ls | lp//prints the names of the file in current directory in 3 columns.

The first command:-

Used to control the width of the output

Default width is 72 character and that can be changed using -w option

Syntax:- $ fmt -w<width>filename

The tr command:-

This command replaces one set of characters with another set

Syntax:- $ tr:'\t' </etc/password

<// redirection symbol used to send contents of /etc/passwd to tr.

13.EDITING AND FORMATTING FILES:

[mca@linux ~]$ cat >names

Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net

[mca@linux ~]$ pr names

2015-06-16 15:08 names Page 1
Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net

[mca@linux ~]$ cat names

Nate nate@engineer.com
Rebecca rif@library.edu
Dan dkraut@bio.ca.edu
Liz liz@thebest.net

[mca@linux ~]$ cat > sample1

this is an example of
a short

file
that contains lines of varying width
[mca@linux ~]$fmt -w 16 sample1
this is an
example of a
short file that
contains lines
of varying
width
[mca@linux ~]$ cat >newfile
bin
robin
dan
[mca@linux ~]$tr '[a-z]''[A-Z]'<newfile
BIN
ROBIN
DAN
[mca@linux ~]$ cat > textfile
command
encodad
lowercase
[mca@linux ~]$ spell textfile
encodad
lowercase
[mca@linux ~]$ispell textfile

Aim:-

To work with date and time

Commands:-

Date command:

'Date' used to print current time and date in any of variety of formats

Also used to set or change the system time.

$date//prints current date,time with default format.

$ date '+'//used to display date with user own formate along with text and using unit symbol.

$ date -d//allows to specify a particular time or date to display

The touch command:-

'touch' used to change the access and modification times for each file.

$ls-l//used to display files with last changed time.

$ls-ul//to display files with last changed time and access time

$touch -m//used to change modification time

$touch -a//used to change access time

14.WORKING WITH DATE AND TIME:

date [OPTION]... [+FORMAT]

1) **To Print current system date and time:**

   [mca@linux ~]$ date

   Sat Nov 10 21:38:15 IST 2017

2) **To print date of next Monday:**

   [mca@linux ~]$ date --date="next mon"

   Sat Nov 10 00:00:00 IST 2017

3) **To display past date**

   [mca@linux ~]$ date --date="1 day ago"

   [mca@linux ~]$ date --date="yesterday"

   Sat Nov 10 21:39:53 IST 2017

4) **To display future date**

   $ date --date="1 day"

   $ date --date="tomorrow"

   Sat Nov 10 21:41:26 IST 2017

5) **To set date:**

$ date -s "Sat Nov 10 21:00:00 PDT 2017"

6) **To display Universal Time:**

[mca@linux ~]$ date -u

Sat Nov 10 16:13:26 UTC 2017

7) **To display Weekday name:**

[mca@linux ~]$ date +%a

[mca@linux ~]$ date +%A

Saturday

8) **To display Month name:**

[mca@linux ~]$ date +%b

mca@linux ~]$ date +%B

November

9) **To display current day of month**:

[mca@linux ~]$ date +%d

08

10) **To display Current Date in MM/DD/YY format:**

[mca@linux ~]$ date +%D

01/08/17

11) **To display date in YYYY-MM-DD format:**

[mca@linux ~]$ date +%F

2017-01-08

12) **To display time as HH:MM:SS, Note: Hours in 24 Format**

[mca@linux ~]$ date +%T

21:47:05

SITAMS	**EXPERIMENT: 15** **AWK UTILITIES**	**Page No.**

Aim:-

To perform Awk commands

Procedure:-

How AWK works:-

AWK is used for pattern matching. This is also called as the pattern scanning language.

Syntax:- $awk '/pattern/{action}'filename.

Default pattern and action:-

This command is used for finding the default pattern and action in the given filename.

Syntax:- $awk '{action}'filenmae

(Or)

$awk 'pattern'/filename

Working with fields:-

This command is used for pattern matching by using fildes.

Syntax:-

$awk '/filename/{action}'filename.

Using standed input and output:-

This command is used for sending the output of one command is given as input to another command.

Syntax:-

$awk '{action}'oldfilename->new filename

Running awk program:-

Awk -f is used to run a program from a file

Syntax:-

$awk -f progfile inputfile

Multiple line programs:-

It is simply consists of multiline pattern location statements

Syntax:-

$awk -f filename1 filename2

Specifying patterns:-

Pattern matching is fundamental part of awk.

Regular expression:-

There are sequence of letters,numbers and special characters that specify string to be matched.

Syntax:-

$awk '/string/{action}'filename

Comparision patterns:-

<,>,<=,>= comparision operators can be used to compare two numbers or two strings.

Syntax:-

$awk '$>10{action}'filename

15.HOW AWK WORKS

[mca@linux ~]$ cat > inventory

awk program was originally developed by Aho,kernighan and Weinberger in 1977.

It is also called as pattern scanning language.

[mca@linux ~]$ awk '/scanning/{print}' inventory **//displays lines that contains the //"scanning" pattern**

It is also called as pattern scanning language.

Default Patterns and Actions

[mca@linux ~]$ cat contacts

Ben	IN	650-333-4321
Dan	AK	907-671-4321
Morissa	NJ	732-741-2431
Robin	CA	650-273-1034

[mca@linux ~]$ awk '{print $1}' inventory **//default pattern**

Ben
Dan
Morissa
Robin

[mca@linux ~]$ awk '/Aho/' inventory **//default action**

Awk program was originally developed by Aho,kernighan and Weinberger in 1977.

Working with Fields

[mca@linux ~]$ cat > contacts

Ben	IN	650-333-4321
Dan	AK	907-671-4321
Morissa	NJ	732-741-2431
Robin	CA	650-273-1034

[mca@linux ~]$ awk '/650-/{print $2}' contacts **//displays 2nd fields that matches the pattern**

IN
CA

Using Standard Input and Output

[mca@linux ~]$ awk '{print}' inventory > standard

[mca@linux ~]$ cat standard

Awk program was originally developed by Aho,kernighan and Weinberger in 1977.

It is also called as pattern scanning language.

Running awk Program from a file

[mca22@linux ~]$ cat > fmtdemo **//awk commands in file fmtdemo**

```
/Mor+/{print $1}
```

[mca@linux ~]$ awk -f fmtdemo contacts **//executing fmtdemo program**

```
Morissa
```

Multiline Programs

[mca@linux ~]$ cat > numberline

```
{
n=n+1
print n " " $0
}
```

[mca@linux ~]$ awk -f numberline contacts //Executing numberline program on contacts

```
1 Ben          IN    650-333-4321
2 Dan          AK    907-671-4321
3 Morissa      NJ    732-741-2431
4 Robin        CA    650-273-1034
```

Specifying Patterns

REGULAR EXPRESSION :

[mca@linux ~]$ cat > stationary1

```
pencil    100   2     3
markers   50    20    22
pens      200   10    12
notes     200   30    34
```

[mca@linux ~]$ awk '/pe*/{print $0}' stationary1

```
pencil  100   2     3
pens    200   10    12
```

Comparision Operators

[mca@linux ~]$ awk '$3 >10 {print $0}' stationary1

```
markers 50    20    22

notes   200   30    34
```

Aim:-

To write a shell script that copis multiple files to a directory.

Procedure:-

Step1:-create a file in vi editor as $vi multidir.

Step 2:-Read the directory name and create a new directory with name as Mkdir
$dir

Step3:-Read the no.of files that we need to add to a directory ‘n’.

Now,read the filename and copy it to the directory at each iteration until ‘n’as cp $filename $dir.

16. SHELL SCRIPT THAT COPIES MULTIPLE FILES TO A DIRECTORY:

$ vi mulindir

```
i=1
echo "Enter New Directory Name"
read dir
mkdir $dir
echo "Enter the Number of Files to be Added"
read n
while [ $i -le $n ]
do
    echo "Enter the file Name"
    read filename
    cp $filename $dir
    let i=$i+1
done
```

output:

```
[mca45@linux ~]$ chmod +x mulindir
[mca45@linux ~]$ ./mulindir
Enter New Directory Name
NewDir
Enter the Number of Files to be Added
2
Enter the file Name
file1
Enter the file Name
file2
[mca45@linux ~]$ cd NewDir
[mca45@linux NewDir]$ ls
file1  file2
[mca45@linux NewDir]$
```

Aim :-

To write a shell script that adds, subtracts, multiplies and divides two integers.

Procedure:-

Step1:- create of file in vi editor as vi arithmetic c.

Step2:- enter option for performing addition, subtracts, multiplies and division and that appears on the screen.

Step3:- Input two integers.

Step4:- using case statements write the arithmetic operation for each case that is for add, sub, div, mul.

Step5:-Use let command at each case for arithmetic expression.

17. SHELL SCRIPT SMALL CALCULATOR (ADDITION, SUBTRACTION, MULTIPLICATION, DIVIDE)

$ vi Arithmetic

```
options "enter -a for addition"
echo "enter -s for subtraction"
echo "enter -m for multiplication"
echo "enter -c for quotient and remainder"
echo "enter the option"
read opt
echo "enter 2 numbers"
read a
read b
case $opt in
      -a)let d=$a+$b
            echo "sum=$d"
            ;;
      -s)let d=$a-$b
            echo "difference=$d"
            ;;
      -m)let d=$a*$b
            echo "product=$d"
            ;;
      -c)let d=$a/$b
        let e=$a%$b
            echo "quotient=$d"
            echo "remainder=$d"
            ;;
*)echo "wrong option"
            ;;
Esac
```

Output:

$ chmod +x Arithmetic
$./Arithmetic
enter -a for addition
enter -s for subtraction
enter -m for multiplication
enter -c for quotient and remainder
enter the option
-a
enter 2 numbers
2
4
sum=6

Aim:-

To write a shell script program that counts the number of lines, words, characters in a file.

Procedure:-

Step1:-create a file in vi editor as

$vi count.

Step2:-Read the file.

Step3:-Count the number of files in the file using wc-d $file.

Step4:-Count the number of files in the file using wc-w $file.

Step5:-Count the number of files in the file using wc-c $file.

18. SHELL SCRIPT COUNTS NUMBER OF LINES AND WORDS IN A FILE:

$ vi count

```
echo "Enter the File Name"
read file
echo " Number of Lines in the file are :"
wc -l $file
echo "Number of Words in the File are :"
wc -w $file
echo " Number of Characters in the File are:"
wc -c $file
```

output:

```
[mca45@linux ~]$ cat sample
The quick brown fox jumped over the lazy dog.
The dog drifted back to sleep and dreamed of bitting the fox
what a foolish sleepy dog
[mca45@linux ~]$
[mca45@linux ~]$ chmod +x count
[mca45@linux ~]$ ./count
Enter the File Name
sample
 Number of Lines in the file are :
3 sample
Number of Words in the File are :
26 sample
 Number of Characters in the File are:
133 sample
[mca45@linux ~]$
```

Aim:-

To write a shell script program that displays the list of all files in the given directory.

Procedure:-

Step1:- read the directory name from the user.

Step2:-List the files under that directory using the ls command ls $path.

19. SHELL SCRIPT THAT DISPLAYS THE LIST OF ALL FILES IN THE DIRECTORY:

$ vi flindir

```
echo " Enter the Directory Name"
read path
echo "List of Files under " $path "are:"
ls $path
```

output:

```
[mca45@linux ~]$ chmod +x flindir
[mca45@linux ~]$ ./flindir
 Enter the Directory Name
NewDir
List of Files under  NewDir are:
file1  file2
[mca45@linux ~]$
```

Aim:-

To write a shell script program to generate multiplication table.

Procedure:-

Step1:-Read the numbers which is required for multiplication table and store in variable 'n'.

Step2:- In for loop write the Expression for multiplication using.

"Let command" let k=$n*$s;

Step3:-print n the multiplication table format at iteration.

20. SHELL SCRIPT TO GENERATE A MULTIPLICATION TABLE:

$ vi mutable

```
echo "Enter Which table you want"
read n
for i in 1 2 3 4 5 6 7 8 9 10
do
        let k=$n*$i
        echo "$n X $i = $k"
done
```

output:

```
[mca45@linux ~]$ chmod +x mutable
[mca45@linux ~]$ ./mutable
Enter Which table you want
3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
3 X 10 = 30
[mca45@linux ~]$
```

Aim:-

To perform mathematical calculations.

Commands :-

Two most powerful and useful unix tools for performing mathematical calculations.

1. BC command
2. DC command

1.**BC command:(basic calculator)**

- Bc used both calculator and mini language for writing mathematical programs.
- $bc.// performs specified mathematical calculations operation.
- Bc command does not save any decimal places in result of a calculation.
- We can save the result of calculation with in a variable.
- Bc command can be used to convert numbers from one base to another.
  I base || to set input base.
  O base || to control outputbase.
- Bc command are also used to define function and that can be used just like built-in-functions.

2.The DC command:(desk calculator)

- Dc is an older alternative to bc.
- Unlike bc, which uses the more familer infix method.
  $dc
  20 10+p|| 20and 10 are added and then p tells dc to print result.
  Q|| is the instruction to quit the program.

21. PERFORMING MATHEMATICAL CALCULATIONS:

Basic Calculator

```
[mca@linux ~]$ bc
32+17
49
Sqrt(49)
7
quit

[mca@linux ~]$ bc
(( (1+5) * (3+4) )/6 )^ 2
49
quit

[mca@linux ~]$ bc –l
scale=6
a(1) * 4
3.141592
quit
```

```
[mca@linux ~]$ bc
ibase=2
11010
26
ibase=1010
quit

[mca@linux ~]$ bc
for( i=1;i<=4;i=i+1) i^2
1
4
6
9
16
quit
```

Desk Calculator

```
[mca@linux ~]$   dc
32  64 + p
96
q

[mca@linux ~]$   dc
1  5+  3  4+  *  6/  2^
49
q
```