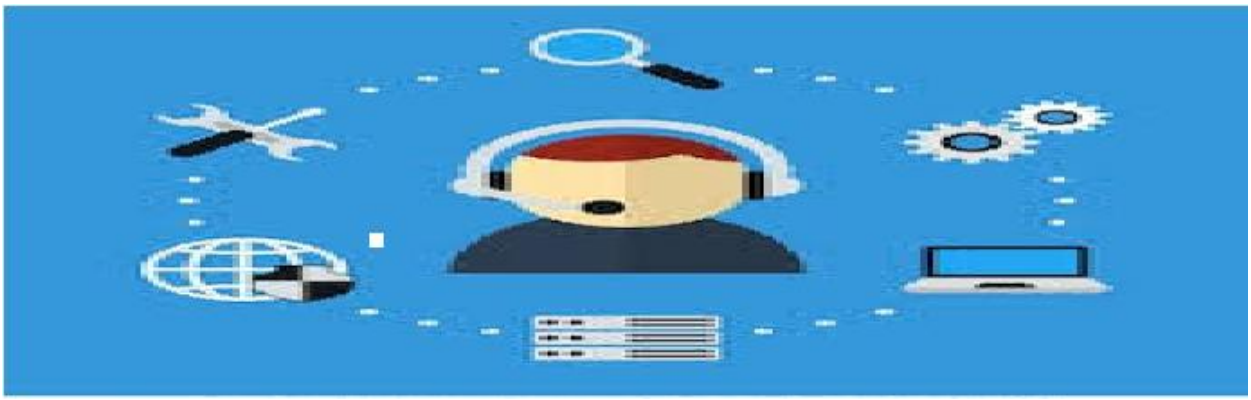
LAB MANUAL



SYSTEM ADMINISTRATION LAB

(FOURTH SEMESTER, COURSE CODE: 4139)
STUDENTS VERSION

Department Of Computer Engineering
Government Polytechnic College, Perumbavoor
KoovappadyP.O,Phone:04842649251email:gptcpbvr@gmail.com

VISION AND MISSION

Government Polytechnic College, Perumbavoor Vision and Mission

Vision

Excel as a centre of skill education molding professionals who sincerely strive for the betterment of society.

Mission

• To impart state of the art knowledge and skill to the graduate and moulding them to be competent, committed and responsible for the well being of society.

• To apply technology in the traditional skills, thereby enhancing the living standard of the community

Department of Computer Engineering

Vision

Excel as a skill centre in Computer Engineering moulding professionals who are adaptive and sincerely strive towards betterment of society

Mission

• To impart state of the art, knowledge ,skill and attitude to the graduates ensuring sustainable development

• To develop adaptiveness for being competent to acquaint with the technological changes.

PROGRAMME OUTCOMES

1. **Basic knowledge:** An ability to apply knowledge of basic mathematics, science and engineering to solve the engineering problems.
2. **Discipline knowledge:** An ability to apply discipline - specific knowledge to solve core and/or applied engineering problems.
3. **Experiments and practice:** An ability to plan and perform experiments and practices and to use the results to solve engineering problems.
4. **Engineering Tools:** Apply appropriate technologies and tools with an understanding of the limitations.
5. **The engineer and society:** Demonstrate knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to engineering practice.
6. **Environment and sustainability:** Understand the impact of the engineering solutions in societal and environmental contexts, and demonstrate the knowledge and need for sustainable development.
7. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
8. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse/multidisciplinary teams.
9. **Communication:** An ability to communicate effectively.
10. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the context of technological changes.

Programme Specific Outcomes (PSOs)
PSO 1: The computer engineering diploma graduate will be able to work in Information Technology industry in the area of development, implementation , testing and maintenance .
PSO 2: The computer engineering diploma graduate will be fit for real time software projects.
PSO 3: A successful career as an entrepreneur with a passion, social commitment and ethical responsibility for real-world applications using optimal resources.

Programme Educational Objectives (PEOs)
PEO-1: To produce technically competent diploma holders in engineering with scientific, analytical, mathematical and problem solving skills.
PEO-2: To develop the habit of quality, safety, self-learning and life-long learning along with environmental awareness.
PEO-3: To equip diploma holders with good management practices, interpersonal skills and entrepreneurial discipline with strong adherence to ethics and values.

INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name , date on which it was done and the page number.

On the right side page of the record following has to be written:

1. Title: The title of the experiment should be written in the page in capital letters.

2. In the left top margin, experiment number and date should be written.

3. Aim: The purpose of the experiment should be written clearly.

4.Apparatus/Tools/Equipments/Components used: A list of the Apparatus/Tools /Equipments /Components used for doing the experiment should be entered.

5. Principle: Simple working of the circuit/experimental set up/algorithm should be written.

6. Procedure: steps for doing the experiment and recording the readings should be briefly described(flow chart/programs in the case of computer/processor related experiments)

7. Results: The results of the experiment must be summarized in writing and should be fulfilling the aim.

8. Inference : Inference from the results is to be mentioned.

On the Left side page of the record following has to be recorded:

1. Circuit/Program: Neatly drawn circuit diagrams/experimental set up.

2. Design: The design of the circuit/experimental set up for selecting the components should be clearly shown if necessary.

3. Observations:

i) Data should be clearly recorded using Tabular Columns.

ii) Unit of the observed data should be clearly mentioned iii) Relevant calculations should be shown. If repetitive calculations are needed, only show a sample calculation and summarize the others in a table.

4. Graphs : Graphs can used to present data in a form that show the results obtained, as one or more of the parameters are varied. A graph has the advantage of presenting large amounts of data in a concise visual form. Graph should be in a square format.

GENERAL INSTRUCTIONS FOR PERSONAL SAFETY

The safety instructions are presented to the attention of the students as a mean of preventing accidents while performing experiments and activities in the communication lab of the department .The purpose is to draw attention to the risks involved in lab activities to prevent human suffering and damage to equipment.

Safety in the laboratory:

- Working in the lab is not allowed without following electricity precautions displayed.
- No individual work is allowed in the lab.
- Laboratory in charge is responsible for the arrangements of your lab activities; Listen carefully to his/her instructions and follow them.

To do and not to do:

- Inform the lab in charge about dangerous conditions and faults in the lab or nearby environment.
- Do not do any action that may harm people or equipments in the lab.
- Do not misuse any of the tools or instruments belong to the lab.
- Strict discipline should be maintained in the laboratory.
- Turn off cell phones before entering the lab.
- At the end and beginning of laboratory, follow 5S procedures and leave the work table clean and tidy.

Electrical Safety:

- Consult Electrical Engineering section available in the campus for electrical safety queries.

- The lab equipment is powered from electrical sockets installed on the tables. Do not use equipment that is powered from a damaged socket.
- Do not use equipment that is powered from flexible cable with damaged insulation or if it's plug is not assembled properly.
- Do not repair or disassemble electrical equipment including replacement of fuses installed in the equipment.
- Do not open the main fuse box, unless it is an emergency and you need to switch off main circuit breaker.
- Be sure to turn off the power and remove the power plug from all equipment before working repairing or assembling.
- Do not plug in or remove equipments while the power is on.
- Emergency Switches:
    - The laboratory has circuit breakers, which is located in the main panel. Identify the place.
    - In an emergency condition, switch off circuit breakers immediately.

Exp No:1 **ROLE OF SYSTEM ADMINISTRATOR**

Problem Statement : To understand the role of system administrator

Description:

The person who is responsible for setting up and maintaining the system or server is called as the system administrator. System administrators may be members of an information technology department. The duties of a system administrator are wide-ranging, and vary widely from one organization to another. Sysadmins are usually charged with installing, supporting, service outages and other problems. Other duties may include scripting or light programming, project management for systems-related projects. The system administrator is responsible for following things:

- User administration (setup and maintaining account)
- Maintaining system
- Verify that peripherals are working properly
- Quickly arrange repair for hardware in occasion of hardware failure
- Monitor system performance
- Create file systems
- Install software
- Create a backup and recovery policy
- Monitor network communication
- Update system as soon as new version of OS and application software comes out
- Implement the policies for the use of the computer system and network
- Setup security policies for users. A sysadmin must have a strong grasp of computer security (e.g. firewalls and intrusion detection systems)
- Documentation in form of internal wiki
- Password and identity management
- Provide required training to all users

RESULT : Familiarization of role of system administrators.

Exp No:2 **LINUX DISTRIBUTIONS**

Problem Statement : To know about different Linux Distributions

A Linux distribution is an operating system made as a software collection based on the Linux kernel and, often, on a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices (for example, **OpenWrt**) and personal computers to powerful supercomputers (for example, **Rocks Cluster** Distribution).

A typical Linux distribution comprises :

- a Linux kernel,
- GNU tools and libraries,
- additional software,
- documentation,
- a window system (the most common being the X Window System),
- a window manager,
- a desktop environment.

Most of the included software is free and open-source software made available both as compiled binaries and in source code form, allowing modifications to the original software. There are **commercially backed distributions**, such as : Fedora (Red Hat), openSUSE (SUSE) and Ubuntu (Canonical Ltd.) Entirely **community-driven distributions**, such as: Debian, Slackware, Gentoo and Arch Linux. Most distributions come ready to use and pre-compiled for a specific instruction set, while some distributions (such as Gentoo) are distributed mostly in source code form and compiled locally during installation.

Popular distributions

Name	Description
Debian	A non-commercial distribution and one of the earliest, maintained by a volunteer developer community
Raspbian	an operating system for the Raspberry Pi
Knoppix	the first Live CD distribution to run completely from removable media without installation to a hard disk, derived from Debian
Ubuntu	a popular desktop and server distribution derived from Debian, maintained by British company Canonical Ltd.
Fedora	Community distribution sponsored by Americancompany .
Red Hat	Red Hat Enterprise Linux a derivative of Fedora, maintained and commercially supported by Red

RESULT :- Familiarization of different Linux distributions

Exp No: 3 **MAN PAGES AND OTHER DOCUMENTS**

Problem Statement : To study about man pages and other documentation provided by Linux

Linux documentation is spread over a number of sources, some of which you will find installed on your system and some of which live out on the net. The most common documentations are :

- Manual pages (man pages), read with the **man** command
- Texinfo documents, read with the **info** command
- HOWTOs, short notes on various subjects (www.tdlp.org)
- Guides, longer treatises on various subjects (www.tdlp.org)
- Distribution-specific documentation
- Web pages associated with specific software projects

The man pages and Texinfo documents constitute the traditional "on-line" documentation.These docs are typically installed with the system; program-specific man pages usually come along for the ride whenever you install a new package.

Organization of the man pages

The Linux man pages are typically divided into nine sections as shown in Table below :

Section	Contents
1	User-level commands and applications
2	System calls and kernel error codes
3	Library calls
4	Device drivers and network protocols
5	Standard file formats
6	Games and demonstrations
7	Miscellaneous files and documents
8	System administration commands
9	Obscure kernel specs and interfaces

The **man** command actually searches a number of different directories to find the manual pages you request. You can determine the search path with the **manpath** command.

$ manpath

If necessary, you can set your **MANPATH** environment variable to override the default path. You can also set the system-wide default in **/etc/man.config** (RHEL and Fedora) or **/etc/manpath.config** (SUSE, Debian, and Ubuntu).

Syntax of man command :

man title (Example : **man ls -** will send the manual pages of **ls** command to the console)

RESULT : - Familiarization of man pages and other documentations provided by Linux.

Exp No: 4 **LINUX FILE SYSTEM HIERARCHY**

Problem Statement : To study about Linux File system hierarchy

Overview of the FHS

Everything in Linux can be reduced to a file. Partitions are associated with files such as /dev/hda1. Hardware components are associated with files such as /dev/modem. Detected devices are documented as files in the /proc directory. The File system Hierarchy Standard (FHS) is the official way to organize files in Unix and Linux directories.

Linux file system and directory structure

Several major directories are associated with all modern Unix/Linux operating systems. These directories organize user files, drivers, kernels, logs, programs, utilities, and more into different categories. The standardization of the FHS makes it easier for users of other Unix-based operating systems to understand the basics of Linux. Every FHS starts with the root directory, also known by its label, the single forward slash (/). All of the other directories shown in Table are subdirectories of the root directory. Unless they are mounted separately, you can also find their files on the same partition as the root directory.

/	The root directory, the top-level directory in the FHS. All other directories are subdirectories of root, which is always mounted on some partition. All directories that are not mounted on a separate partition are included in the root directory's partition.
/bin	Essential command line utilities. Should not be mounted separately**;**
/boot	Includes Linux startup files, including the Linux kernel. Can be small; 16MB is usually adequate for a typical modular kernel. If you use multiple kernels, such as for testing a kernel upgrade, increase the size of this partition accordingly.
/etc	**Most basic configuration files.**
/dev	Hardware and software device drivers for everything from floppy drives to terminals. Do not mount this directory on a separate partition.
/home	Home directories for almost every user.

/lib	Program libraries for the kernel and various command line utilities. Do not mount this directory on a separate partition.
/mnt	The mount point for removable media, including floppy drives, CD-ROMs, and Zip disks.
/opt	Applications such as WordPerfect or StarOffice.
/proc	Currently running kernel-related processes, including device assignments such as IRQ ports, I/O addresses, and DMA channels.
/root	The home directory of the root user.
/sbin	System administration commands. Don't mount this directory separately.
/tmp	Temporary files. By default, Red Hat Linux deletes all files in this directory periodically.
/usr	Small programs accessible to all users. Includes many system administration commands and utilities.
/var	Variable data, including log files and printer spools.

Types of Files Used by Linux

When working with Linux, you need to be aware of the fact that there are a number of different file types used by the file system. This is another area where the Linux file system differs significantly from the Windows file system. With a Windows file system you basically have two entry types in the file system:

- Directories
- Files

With Linux, there are a variety of different file types used by the file system. These include the file types shown in Table

File Type	Description
Regular Files	These files are similar to those used by the file systems of other operating systems—for example, executable files, OpenOffice.org files, images, text configuration files, etc.
Links	These files are pointers that point to other files in the file system.
FIFOs	FIFO stands for First In First Out. These are special files used to move data from one

	running process on the system to another. A FIFO file is basically a queue where the first chunk of data added to the queue is the first chunk of data removed from the queue. Data can only move in one direction through a FIFO.
Sockets	Sockets are similar to FIFOs in that they are used to transfer information between sockets. With a socket, however, data can move bi-directionally.

RESULT : Familiarization of file system hierarchy in Linux

Exp No:5 **ABOUT LINUX SHELL**

Problem Statement: To understand about Linux command line / shell

The GNU/Linux shell is a special interactive utility. It provides a way for users to start programs, manage files on the file system, and manage processes running on the Linux system. The core of the shell is the command prompt. The command prompt is the interactive part of the shell. It allows you to enter text commands, interprets the commands, then executes the commands in the kernel.

The shell contains a set of internal commands that you use to control things such as copying files, moving files, renaming files, displaying the programs currently running on the system, and stopping programs running on the system. Besides the internal commands, the shell also allows you to enter the name of a program at the command prompt. The shell passes the program name off to the kernel to start it.

There are quite a few Linux shells available to use on a Linux system. Different shells have different characteristics, some being more useful for creating scripts and some being more useful for managing processes. The default shell used in all Linux distributions is the **bash** shell. The bash shell was developed by the GNU project as a replacement for the standard Unix shell, called the Bourne shell (after its creator). The **bash** shell means ‘‘**Bourne again shell**’’. Besides the **bash** shell , thera are several other popular shell as mentioned below :

ash	A simple, lightweight shell that runs in low-memory environments but has full compatibility with the bash shell
korn	A programming shell compatible with the Bourne shell but supporting advanced programming features like associative arrays and floating-point arithmetic
tcsh	A shell that incorporates elements from the C programming language into shell scripts
zsh	An advanced shell that incorporates features from bash, tcsh, and korn, providing advanced programming features, shared history files, and themed prompts

Getting shell Interface

There are several ways to get shell interface in Linux. Theree of the most common are :

- The shell prompt
- Terminal window
- Virtual console

Shell Prompt -If your Linux system has no GUI, you will get a shell prompt after you login. The default prompt for a regular user is a dollar sign : **$.**The default prompt for the root user is a pound (hash) sign: **#**

In most Linux systems, the $ and # prompts are preceded by your username, system name and current directory name.

Terminal Window

From ther desktop GUI, we can open a terminal emulator program (terminal window) to start a shell.

Right click the desktop and choose **open in Terminal** or use the shortcut **Ctrl-Alt-T**

Virtual consoles

Virtual consoles are a way to have multiple shell sessionsopen at once in addition to the GUI.

To switch between one of seven virtual consoles , **Ctrl-Alt-F1** (F2,F3,F4,F5,F6)

When you are finished, type **exit** to exit the shell

To find all available shells in your system type following command:

$ cat /etc/shells

To find your current shell type following command

$ echo $SHELL

Basic Shell Commands

cp	copy files and directories
mv	move or rename files and directories
rm	remove files and directories
mkdir	create directories
chmod	modify file access rights
su	temporarily become the superuser
chown	change file ownership
chgrp	change a file's group ownership
ps	list the processes running on the system
kill	send a signal to one or more processes (usually to "kill" a process)
jobs	an alternate way of listing your own processes
bg	put a process in the background
fg	put a process in the foreground

RESULT :- Familiarization of fundamentals of Linux shell

Exp No:6 **FILE & DIRECTORY COMMANDS**

Problem Statement : To learn basic file & directory commands

Command:**ls**

- Lists the contents of current working directory.

Syntax:

- **ls [options] [file]**
- Options :

-l - list the files in the long format
-a - list all entries, including the hidden files
-d - list the directory files instead of its contents
-t - lists in order of last modification time

(**man ls** to get details of all options)

Wild cards or meta characters used to represent filenames

* Matches any string or group of characters.
? Matches any single character.
[...] Matches any one of the enclosed characters
[..-..] A pair of characters separated by a minus sign denotes a range.

Examples :

ls *	will show all files
ls -l a*	will show all files whose first name is starting with letter 'a'
ls -l *.c	will show all files having extension .c
ls ut*.c	will show all files having extension .c but file name must begin with 'ut'
ls ?	will show all files whose names are 1 character long
ls fo?	will show all files whose names are 3 character long and file name begin with fo
ls [abc]*	will show all files beginning with letters a,b,c
ls /bin/[a-c]*	Will show all files name beginning with letter a,b or c in the folder /bin

Command:cat

- Used to show the file contents on standard output, create disk files from standard input,concatenate files etc.

Syntax: cat [files]

Options : (**man cat** to get details of all options)

Examples:

cat abc.txt	Display the content of file abc.txt on standard output device
cat > def.txt **type the file contents^D to save**	Save the typed contents to file named def.txt
cat abc.txt def.txt	Display the content of both files

cat >> abc.txt **type the contents to be** **added** **^D to save**	Will append the entered text to abc.txt
cat abc.txt def.txt > pqr.txt	eate pqr.txt by merging the contents of abc.txt & def.txt

Command:mkdir

- Creates the directorie(s) specified

Syntax: mkdir directorie(s)

Options :

-p - no error if exists, makes parent directories as needed

-v - print a message for each created directory

(**man mkdir** to get details of all options)

Examples:

mkdir s6ct	Creates the folder or directory s6ct in the current folder
mkdir sa np prj	Creates three folders

Command : **cd**

- Change the directory

Syntax: cd directory path

Options : (**man cd** to get details of all options)

Examples:

cd s6ct	Change to the subdirectory s6ct from its parent
cd ~	Change to the home folder
cd ..	Change to the parent folder
cd /	Chnage to the root folder
cd ~/Documents	Change to the Documents folder under the home directory
cd ~/Desktop/s6ct	Change to the folder s6ct under the Desktop folder of current user

Command : **rmdir**

- Remove directories if they are empty

Syntax: rmdir directorie(s)

Options : -p - remove directory and its ancestors

(**man cd** to get details of all options)

Examples:

rmdir s6ct	Deletes the folder s6ct
rmdir ~/a/b/c	Deletes the folder c, its parent b and the parent of b from the home folder
rmdir ~/a/b/c ~/a/b ~/a	Deletes the folder c, its parent b and the parent of b from the home folder

Command : pwd

- print name of current/working directory

Syntax: pwd

Options : (**man pwd** to get details of all options)

Command : cp

- Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Syntax: cp source(s) directory

cp source destination

Options : (**man cp** to get details of all options)

Examples:

cp abc.txt xyz.txt	Copy the file abc.txt to another file xyz.txt
cp abc.txt xyz.txt test	Copy the files abc.txt & xyz.txt to the folder test

Command : mv

- Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

Syntax: mv source(s) directory

mv source destination

Options : (**man cp** to get details of all options)

Examples:

mv abc.txt 123.txt	Rename the file abc.txt as 123.txt
mv abc.txt xyz.txt test	move the files abc.txt & xyz.txt to the folder test

Command : rm

- remove files or empty directories

Syntax: rm [option] file

Options :

-i prompt before every removal

-d remove empty directory

(**man rm** to get details of all options)

Examples:

rm -i abc.txt	Remove or delete the file abc.txt
rm -d test	Delete the folder test, provided it is empty

RESULT : Familiarization of basic commands for managing files and folders.

Exp No:7 **USING STREAMS, PIPES, FILTERS AND REDIRECTION**

Problem Statement : To Combine multiple commands to do tasks using streams, pipes, filters and redirection

Connecting and Expanding Commands

- A powerful feature of the shell is the capability to redirect the input and output of commands to and from other commands and files
- The following metacharacters are used to connect commands

Metacharacter	Description
|	It is the pipe character which connects the output from one command to the input of another command
>	It is a redirection operator which directs the output of a command to a new file
>>	It appends an existing file with the output of a command
< <	redirects the content of a file as input to a command

Eg: **less < filename {}** Curly braces can be used as expansion characters

Eg : To create a set of **empty** files such as **memo1, memo2, ... , memo5**, use the command :

$ touch memo{1,2,3,4,5} $ ls

Command : grep

- print lines matching a pattern
- grep searches the named input FILEs for lines containing a match to the given PATTERN. If no files are specified, or if the file "-" is given, grep searches standard input.
- Syntax: **grep** [OPTION]... PATTERN [FILE]...

Options : **-i** - ignore case while matching

-v - to select non-matching lines

-w - search for whole words

-x - matches the whole line

-c - print a count of matching lines

-n - prefix each line of output with line number

(**man grep** to get details of all options)

Examples

grep ee names	Search for the pattern **ee** in the file **names**
grep -v -n ee names	Search for lines not containing the pattern **ee**, also print the line numbers
ls | grep -c pdf	Print the count of lines containing the word **pdf** in the output of **ls** command
ls | grep pdf > test.txt	Creates the file **test.txt** containing the output of **ls | grep pdf**
cat /etc/passwd | grep student	Search for the username **student** in the system password file **/etc/passwd**

Command : cut

- Print selected parts of lines from each FILE to standard output.
- Syntax: **cut** OPTION... [FILE]...
- Options : **-c** - cut the character in the specified position of each line from a file

  -d - specifies the field delimiter

  -f - specify the field to be selected

  -s - do not print lines not containing delimiters

(**man cut** to get details of all options)

Examples

cut -c4 names	Selects only the 4th character from each line of the file **names**
cut -c2-4 names **cut -c-4 names** **cut -c2- names**	Selects the range of characters 2-4 from each line of the file Selects the range of characters 1-4 from each line of the file Selects the all characters from 2nd position from each line of the file
cut -d';' -f1,2 names	Selects the first and second fileds delimited by ; from the file **names**
cat /etc/passwd | grep student | cut -d':' f6	Display the 6th field (home directory of user **student**) from passwd file

Command : wc

- Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. With no FILE, or when FILE is -, read standard input.
- Syntax: **wc** [OPTION]... [FILE]...
- Options : **-c** - print the bytes count
  -m - print the character count
  -l - print the newline counts
  -w - print the words counts

(**man wc** to get details of all options)

Examples

wc names	Display the count of lines, words & characters in the file, **names**
wc -w names	Display only the count of words along with the file name
ls | wc -l	Display the number of lines in the output of **ls** command

Command : more

- more is a filter for paging through text one screenful at a time.
- Syntax **:more** [options] file...
- Options : (**man more** to get details of all options)

Examples

more test1.txt	Page by page forward display of the content of the file, **test1.txt**
wc -w names	Display only the count of words along with the file name
ls -l | more	Page by page display of directory listing in long format

Command : less

- less is a program similar to more, but it has many more features.
- Syntax **: less** [options] file...

- Options : (**man less** to get details of all options)

Examples

less test1.txt	Page by page scrollable display of the content of the file **test1.txt**
wc -w names	Display only the count of words along with the file name
ls -l | less	Page by page (scrollable)display of directory listing in long format

Difference between more & less

less is a full-screen application that gives you a searchable, scrollable window and clears the screen after you exit, less can also be backgrounded and restored like other full screen terminal applications. less also has a command to open the currently viewed file in your default editor. **more** just prints the text as is, stopping for page breaks, and does not clear the screen, it can be backgrounded but it doesn't clear the screen. more also only reads in the file as it displays where less may read the file into memory first. each handles line wrapping differently which gives different results when selecting and pasting text.

Command : head

- Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input. Syntax **: head** [OPTION]... [FILE]...

- Options : **-c[-]K** - print the first K bytes of each file; with the leading '-', print all but the last K bytes of each file

  -n[-]K - print the first K lines instead of the first 10; with the lead ing '-', print all but the last K lines of each file

  (**man head** to get details of all options)

  Examples

head test1.txt	Display only the first 10 lines of the file **test1.txt**
ls | head -n2	Display the first 2 lines of **ls** command
ls | head -n-2	Display all lines of **ls** command except the last 2 lines

Command : tail

- Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input. Syntax **: tail** [OPTION]... [FILE]...

- Options : **-cK** - output the last K bytes; or use -c +K to output bytes starting with the Kth of each file

  -nK - output the last K lines, instead of the last 10; or use -n +K to output starting with the Kth

  -n[-]K - print the first K lines instead of the first 10; with the lead ing '-', print all but the last K lines of each file

  Examples

tail test1.txt	Display only the last 10 lines of the file **test1.txt**
ls | tail n2	Display the last 2 lines of **ls** command
ls | less -n+30	Display all lines of **ls** command except the first 29 lines
ls | head -n5 | tail -n1	Displays the 5th line of **ls** command

Command : sort

- sort lines of text files Write sorted concatenation of all FILE(s) to standard output.
- Syntax **: sort** [OPTION]... [FILE]...

- Options : **-b** -ignore leading blanks

  -f -ignore case

  -n -numeric-sort, compare according to string numerical value

  -r -reverse order sorting

Examples

sort names	Sort the lines of text in the file **names**
sort -r names	Sort the lines of text in the file **names** in reverse order
sort numbers	Sort the numeric values in the file **numbers** in alphanumeric order
sort -n numbers	Sort the numeric values in the file **numbers** in numerical order
ls : sort	Sorted display of directory listing

Command : unique

- report or omit repeated lines Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).
- Syntax **: uniq** [OPTION]... [INPUT [OUTPUT]]

- Options : **-c** - prefix lines by the number of occurrences

  -d - only print duplicate lines

  -i - ignore-case

  (**man unique** to get details of all options)

Examples

sort names | uniq	Filters the duplicates of the file , **names**
sort names | uniq -c	Display the lines of the file, **names** , with number of occurrences
sort names | uniq -d -c	Display only the duplicate lines with number of occurrences

RESULT : Familiarization of combining multiple commands to perform tasks using streams, pipes, filters and redirection

Exp No:8 **SHELL, SHELL SCRIPTS & BASIC PROGRAMMING CONSTRUCTS**

Problem Statement : To learn about Linux shell & the programming language constructs used in shell scripts.

What is a shell?

- The shell is a command language interpreter in an operating system. It is a program that takes commands typed by the user and calls operating system services to execute those commands. Shell act as the interface between user and the OS kernel.
- There are different kinds of shell programs available with Linux OS, each having its own commands and functions.
- Major types of shells are :
    - Bourne shell (default prompt is $ character)
    - Various Bourne shells are :
        - Bourne shell (sh) (Written by Steve Bourne)
        - Korn shell (ksh)
        - Bourne Again shell (bash)
    - C shell (default prompt is % character)
        - Various C-type shells are :
            - C shell (csh)
            - TC shell (tcsh)
- **bash** is the most popular shell in the open source community. It is the default shell used with Linux OS. It is the executable file stored in **/bin** folder
- To see the default shell , display the environment variable SHELL

    echo $SHELL

Shell Script

- A shell script is a file that contains a sequence of commands for an OS that are to be executed together like a program. A shell script is usually created for command sequences for which a user has a repeated need.

How to create & execute shell scripts ?

- To create scripts, use any one of the editor programs like **gedit** and save the script command sequence with extension **.sh**
- To execute the script , there are **two** methods :

Method 1 : Change the mode of the script file as executable using **chmod** command

chmod 777 scriptname.sh

OR

chmod 755 scriptname.sh

Then run the script **./scriptname.sh**

Method 2 : Run the script by specifying the shell along with the script file (No need to make it executable)

bash scriptname.sh

Result: Familiarization of basics of shell.

Exp No:9 **PRINT HELLO**

Problem Statement: Write a shell Script to print the phrase “Hello World”

Theory

Variables in Shell Scripts

• Information to be reused later can be stored in variables

• Variable names in shell scripts are case sensitive.

Syntax of variable definition

variable_name = value

• To substitute the value of variables, prefix the variable with **$** sign

• The output of a command or the result of an expression can also be saved in variables.

Enclose the command in parentheses and prefix it with **$** symbol or enclose the command in backticks (`) .

▪ **variable_name = $(command)**

▪ **variable_name = `command`**

Procedure

1. Assign string "Hello World" to variable msg

2. Print value of msg using command echo

Source Code

****

Sample Output

Hello World

Result :

****

Exp No:10 **Print system information**

Problem Statement: Write a shell Script to print system information.

Source Code

****

Result:

****

Exp No:11 **SUM and AVERAGE**

Problem Statement: Write a shell Script to calculate the sum and average of 4 integers

Theory

read command : Used to read values to variables interactively from standard input during runtime.

read -p "prompt message " var1 var2 .....

Performing arithmetic in shellscripts

Integer arithmetic can be performed using :

- built-in **let** command
- **expr** command
- **bc** command (It is a calculator application that is available in most linux distributions)

- **Example :-** To evaluate 12 / 4 and store the result in variable **a** , the three methods are as follows :

let a=12/4

a = `expr 12 / 4`

a=`echo " 12 / 4 " | bc`

Procedure

1. Read values of a,b,c,d
2. Calculate sum =a+b+c+d and average =sum/4
3. Print average.

Source Code

****

Sample Output

Enter 4 numbers :

2 4 6 8

sum =20

Average=5.0

Result: The script has executed with inputs and output verified.

Exp No:12 **Menu Driven Calculator**

Problem Statement: Write a shell script to perform arithmetic operations based on menu.

Theory

case command -It is similar to switch statement in c-programming

```
case $var
in
Result1)
        Statement group-1 ;;
Result2)
        Statement group-2 ;;
        ...
*)
    Default Statement group ;;
esac
```

Procedure

1.Read two numbers

2.Enter the choice of arithmetic operation

3.Define case with arithmetic operations

4.Repeat step 1 to 3 again until choice for exit is given

Source Code

****

Sample Output

```
Enter first no:
5
Enter second no:
10
1.Addition
```

2.Subtraction
3.Multiplication
4.Division
Enter your choice1
Sum =15

Do u want to continue?

Result:

****

Exp No: 13 **Decrementing N upto zero**

Problem Statement: Write a shell script to accept a number N and decrement N upto zero.

Theory

Syntax of while loop

```
while [ condition ]
do
        statement1
        ……
        statementN
done
```

Procedure

1. Start
2. Read a value N
3. Print N
4. Decrement N by 1
5. Repeat step 2 and step3 until N reaches zero
6. Stop

Source Code

Sample Output

Enter value for N

5

…….
5
4
3
2
1

Result:

****

Exp No:14 **Menu driven program to perform some tasks**

Problem Statement: Write a shell script to list directory, present working directory name, print system date and clear the screen .

Theory

Commands

ls -to list directory

pwd -to print present working directory name

date - system date

clear - clear the screen

Procedure

1. Start
2. Print options to list directory, present working directory name, print system date and clear the screen
3. Read choice
4. Print the details according to the option
5. Repeat step 3 and step4 until the entered option is exit
6. Stop

Source Code

****

Sample Output

1. List directory

2. Print Directory Name

3. Print system Date

4. Clear the screen

5. Exit

Your Choice :3

Tue Feb 5 16:00:08 @ST 2019

Press Any Key to continue ....

Result

The script has executed with inputs and output verified.

Exp No:15 **Command line arguments**

Problem Statement: Write a script to read and print four command line arguments

Theory

Inputs needed to execute a command or script can be provided from command line

command arg1 arg2 .....

The special variable **$0, $1, $2, $3 ..**.etc can be used inside the script to access the command as well as the arguments mentioned with the command

Procedure

1.Start

2.Write commands $#,$0,$* for argument count,command and all arguments respectively.

3.Display arguments

4.Stop

Source Code

****

SampleOutput

Run script-./arg.sh hello how are you

Argument count :4

The command: ./arg.sh

All arguments: hello how are you

Argument 1: hello

Argument 2: how

Argument 3: are

Argument 4: you

Result

****

Exp No:16 **File existence and file type using command line**

Problem Statement: Write a script program to check the given argument is a file or not.

Theory

Syntax if-elif-else

If [condition];then Stmt1 …….. stmtN

elif [condition]then Stmt1 …….. stmtN

…..

elif [condition]then Stmt1 …….. stmtN

else Stmt1 …….. stmtN

fi

Procedure

1.Start

2.Check whether file name entered or not by $#

3.If $1 is directory or file print it is directory or file respectively

4.Otherwise print object does not exists

5.Stop

Sample Output

Run script-./filecheck.sh file.sh

file.sh is a file

Result

****

Exp No:17 **Sorting of numbers**

Problem Statement: Write a shell script to sort N numbers in ascending order

Theory

Syntax of for loop

for((initialization;condition checking;increment))

do

stmts

done

Procedure

1.Start

2. Enter Array size n and read n elements

3. Print the entered elements

4. Set i=1

5.Check i<=n.if yes set j=1 otherwise go to step 8

6. check j<=n-i.if not go to step 5

7.if a[j]>a[j+1] interchange both. Increment j and go to step 6

8.Print array

9.Stop

Source Code

****

Enter Array size : 5

Enter Number $i :3 23 5 10 17

Original Numbers

---

3 23 5 10 17

Sorted Numbers

3 5 10 17 23

Result

****

Exp No:18 **Linear Search**

Problem Statement: Write a shell script to search for an element within an array using linear search

Theory

Array

Accessing array element- ${a[i]}

Procedure

1.Start
2.Enter size of array
3. Read elements of array a[]
4. Enter value to search,s

5.Set i=1,flag=0

6.If a[i] equal to s ,set flag to 1 and go to step 7. Otherwise increment i and repeat step 6

7. if flag equal to 1 ,print s is found at I,otherwise print not found

8.Stop

Source Code

****

Sample Output
Enter array size:4
Enter 4 elements:
6
9
2
3
Enter Search element:2
2 is found at 3

Result
****

Exp No:19 **Binary Search**

Problem Statement: Write a shell script to search for an element within an array using binary search

Theory

Array

Accessing array element- ${a[i]}

Procedure

1.Start

2.Enter size of array

3. Read elements of array a[]

4. Enter value to search,s

5. Set low=1 high=$n flag=0

6.If low is less than or equal to high ,calculate mid=low+high/2

7. if a[mid] >s,set low=mid+1

8.If a[mid]<s,set high=mid-1.

9.if a[mid]==s, set flag=1 and go to step 9

8.increment low and go to step 6

9.if flag==1,print found otherwise print not found

10.Stop

Source Code

****

Sample Output
Enter array size:4
Enter 4 elements:
60
65
72

83
Enter Search element:72
72 is found at 3

Result

****

Exp No:20 **Pattern Search**

Problem Statement: Write a shell script to search for a given pattern (string) within the content of a given file.

Theory

Command : grep

- print lines matching a pattern
- grep searches the named input FILEs for lines containing a match to the given PATTERN. If no files are specified, or if the file "-" is given, grep searches standard input.
- Syntax: **grep** [OPTION]... PATTERN [FILE]...

Options : **-i** - ignore case while matching

-v - to select non-matching lines

-w - search for whole words

-x - matches the whole line

-c - print a count of matching lines

-n - prefix each line of output with line number

Procedure

1.Start
2.Read file name
3. Read string to search
4. Find the occurrence of string using command grep
5. Stop

Source Code

****

Sample Output

enter file name :file1.txt
Enter search string :hello
Contents of file1.txt
hello,this is an example
1 occurrence of hello found in file1.txt

Result

****

Exp No:21 **Prime From Fibonacci Series**

Problem Statement: Write a shell script to print prime numbers from Fibonacci series

Procedure

1.Start
2.Read limit N
3. f1=1,f2=1,r=1,f=f1+f2
4.
5. Stop

Source Code

****

Sample Output

Enter N
7
Prime Numbers from Fibo Series < $n
2
3
5

Result

****

Exp No:22 **Print the given number of lines of a file**

Problem Statement: Write a shell script to count the number of characters, words and lines within a specified file.

Theory

Command : wc

- Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. With no FILE, or when FILE is -, read standard input.
- Syntax: **wc** [OPTION]... [FILE]...
- Options : **-c** - print the bytes count
  -m - print the character count
  -l - print the newline counts
  -w - print the words counts

Procedure

1. Start
2. Read file name
3. Display its contents
4. Display its line count, character count and line count by using wc command
5. Read n for number of lines to display and print using head command
6. Stop

Source Code

****

Sample Output

Enter a filename

File.sh
Contents

Gptc perumbavoor
Computer Engineering
Character count in file File.sh36
Word count in file File.sh4
Line count in file File.sh2

Enter no of lines to display
1
Gptc perumbavoor

Result

****

Exp No:23 **CASE CONVERSION**

Problem Statement: Write a shell script to convert all lower case letters in a file to upper case letters (Use **tr** command)

Procedure

1.Start
2.Read file name
3. Print its contents
4. convert lower case to upper using tr command and write it to a new file
5. Print new file contents
6.Stop

Source Code

****

Sample Output

```
enter file name :hello.txt
Contents of Original File hello.txt
----------
Gptc perumbavoor
New file with upper case alphabets
----------
GPTC PERUMBAVOOR
```

Result

****

****