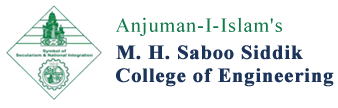
UNIX & SHELL PROGRAMMING LAB MANUAL (II YEAR – I SEMESTER)

(2024-2025)



**Prepared by**

**DR.SHABEENA SAYED**

**Assistant Professor**

**IT**

***PROGRAMOUTCOMES(POs)***

Engineeringstudentswillbeableto:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **Design/development ofsolutions:** Designsolutions for complex engineeringproblems and designsystemcomponents or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations ofcomplex problems:** Useresearch-based knowledgeand research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Applyreasoning informedbythecontextual knowledgetoassess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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.
12. **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.

**PROGRAMSPECIFICOUTCOMES**

**Studentswillbeableto**

**13.**

**PSO1**:Organize,maintainandprotectITInfrastructuralresources.

**PSO2**:DesignandDevelopweb,mobile, andsmartappsbasedsoftwaresolutionstothereal

**CONTENTS**

[5]

|  |  |  |
| --- | --- | --- |
| **S.No** | **ListofExperiments** | **Page No** |
| 1 | TO STUDY INTRODUCTION OF UNIX AND DO THE INSTALLATION. |  |
| 2 | Execution of Unix General Purpose Utility Commands like echo, clear, exit, date, time, uptime, cal, cat, tty, man, which, history, id, pwd, whoami, ping, ifconfig, pr, lp, lpr, lpstat, lpq, lprm, cancel, mail, etc. |  |
| 3 | A.  Study of Unix file system (tree structure), file and directory permissions, single and multiuser environment  B.  Execution of File System Management Commands like ls, cd, pwd, cat, mkdir, rmdir, rm, cp, mv, chmod, wc, piping and redirection, grep, tr, echo, sort, head, tail, diff, comm, less, more, file, type, wc, split, cmp, tar, find, vim, gzip, bzip2, unzip, locate, etc. |  |
| 4 | Execution of User Management Commands like who, whoami, su, sudo, login, logout, exit, passwd, useradd/adduser, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, chown, chage, chgrp, chfn, etc |  |
| 5 | ToPracticeonprocess,disk,networkutilities |  |
| 6 | **To practice Memory Management Commands** in UNIX/Linux with **description**, **syntax**, and **examples** |  |
| 7 | TopracticeUNIXcommandsonViEditor. |  |
| 8 | a) Study of Shell, Types of Shell, Variables and Operators  b) Execute the following Scripts (at least 6):  (i) Write a shell script to perform arithmetic operations.  (ii) Write a shell script to calculate simple interest.  (iii) Write a shell script to determine largest among three integer numbers.  (iv) Write a shell script to determine a given year is leap year or not  (v) Write a shell script to print multiplication table of given number using while statement.  (vi) Write a shell script to search whether element is present is in the list or not.  (vii) Write a shell script to compare two strings.  (viii) Write a shell script to read and check if the directory / file exists or not, if not make the directory / file.  (ix) Write a shell script to implement menu-driven calculator using case statement.  (x) Write a shell script to print following pattern  \*  \* \*  \* \* \*  \* \* \* \* |  |
| 9 | Execute following script using grep/sed command  (i) Write a script using grep command to find the number of words character, words and lines in a file.  (ii) Write ascriptusing egrep command to display list of specific type of files in the directory.  (iii) Write a script using sed command to replace all occurrences of particular word in given a file.  (iv) Write a script using sedcommand to print duplicated lines in input. |  |
| 10 | Execute following script using Awk/perl command  (i) Write an awk script to print all even numbers in a given range.  (ii) Write an awk script to develop a Fibonacci series (take user input for number of terms).  (iii) Write a perl script to sort elements of an array.  (iv) Write a perl script to check a number is prime or not. |  |
| 11 | Assignment 1 |  |
| 12 | Assignment2 |  |

1. TO STUDY INTRODUCTION OF UNIX AND DO THE INSTALLATION.

2. Execution of Unix General Purpose Utility Commands like echo, clear, exit, date, time, uptime, cal, cat, tty, man, which, history, id, pwd, whoami, ping, ifconfig, pr, lp, lpr, lpstat, lpq, lprm, cancel, mail, etc.

### 1. echo

The echo command is used to display a line of text/string on the terminal.

**Syntax:**

echo [options] [string]

**Example:**

echo Hello, World!

### 2. clear

The clear command clears the terminal screen.

**Syntax:**

clear

**Example:**

clear

### 3. exit

The exit command is used to exit the current shell or terminal.

**Syntax:**

exit

**Example:**

exit

### 4. date

The date command displays the current date and time.

**Syntax:**

date [options]

**Example:**

date

### 5. time

The time command is used to determine how long a command takes to run.

**Syntax:**

time [command]

**Example:**

time ls

### 6. uptime

The uptime command displays how long the system has been running.

**Syntax:**

uptime

**Example:**

uptime

### 7. cal

The cal command displays a calendar.

**Syntax:**

cal [month] [year]

**Example:**

cal

cal 12 2025

### 8. cat

The cat command concatenates and displays files.

**Syntax:**

cat [file]

**Example:**

cat file.txt

### 9. tty

The tty command prints the file name of the terminal connected to standard input.

**Syntax:**

tty

**Example:**

tty

### 10. man

The man command displays the manual pages for other commands.

**Syntax:**

man [command]

**Example:**

man ls

### 11. which

The which command locates a command.

**Syntax:**

which [command]

**Example:**

which ls

### 12. history

The history command lists previously executed commands.

**Syntax:**

history

**Example:**

history

### 13. id

The id command prints real and effective user and group IDs.

**Syntax:**

id

**Example:**

id

### 14. pwd

The pwd command prints the current working directory.

**Syntax:**

pwd

**Example:**

pwd

### 15. whoami

The whoami command prints the username of the current user.

**Syntax:**

whoami

**Example:**

whoami

### 16. ping

The ping command tests the network connection to a server.

**Syntax:**

ping [host]

**Example:**

ping google.com

### 17. ifconfig

The ifconfig command configures network interfaces.

**Syntax:**

ifconfig

**Example:**

ifconfig

### 18. pr

The pr command prepares files for printing.

**Syntax:**

pr [options] [files]

**Example:**

pr file.txt

### 19. lp, lpr

The lp or lpr command submits print jobs.

**Syntax:**

lp [options] [files]

**Example:**

lp file.txt

### 20. lpstat

The lpstat command displays printer status information.

**Syntax:**

lpstat [options]

**Example:**

lpstat -p

### 21. lpq

The lpq command displays print queue status.

**Syntax:**

lpq [options]

**Example:**

lpq

### 22. lprm

The lprm command removes print jobs from the print queue.

**Syntax:**

lprm [job ID]

**Example:**

lprm 123

### 23. cancel

The cancel command cancels print jobs.

**Syntax:**

cancel [job ID]

**Example:**

cancel 123

### 24. mail

The mail command sends and receives emails.

**Syntax:**

mail [options] [recipient]

**Example:**

mail -s "Subject" user@example.com

3.

A.Study of Unix file system (tree structure), file and directory permissions, single and multiuser environment

B.

Execution of File System Management Commands like ls, cd, pwd, cat, mkdir, rmdir, rm, cp, mv, chmod, wc, piping and redirection, grep, tr, echo, sort, head, tail, diff, comm, less, more, file, type, wc, split, cmp, tar, find, vim, gzip, bzip2, unzip, locate, etc.

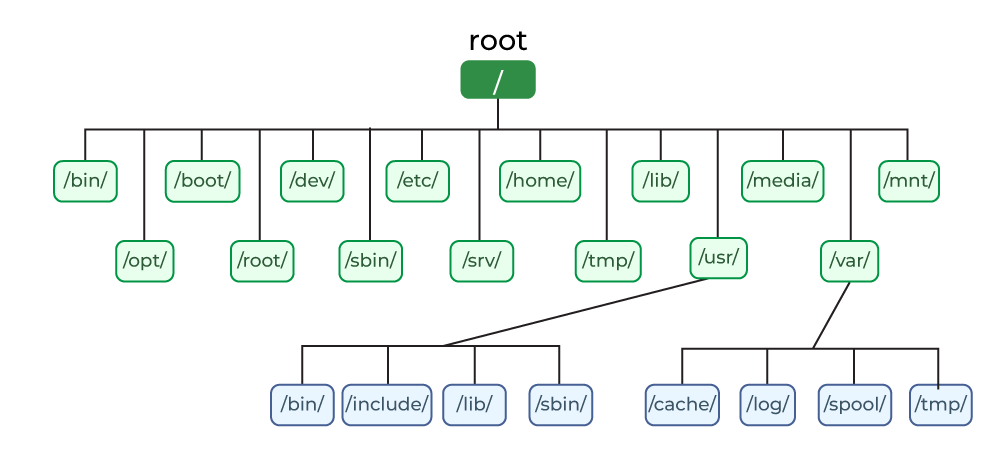
.

**Aim:**Toimplement filepermissions

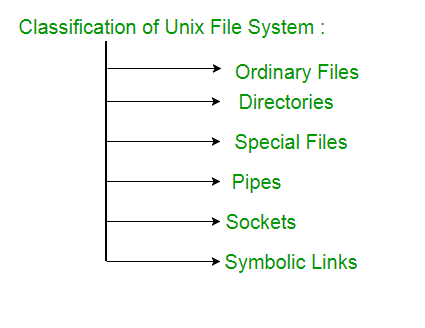
**Description:**

| **NAME** | **DESCRIPTION** |
| --- | --- |
| **/** | | The slash / character alone denotes the root of the filesystem tree. |
| **/bin** | | Stands for “binaries” and contains certain fundamental utilities, such as ls or cp, which are generally needed by all users. |
| **/boot** | | Contains all the files that are required for successful booting process. |
| **/dev** | | Stands for “devices”. Contains file representations of peripheral devices and pseudo-devices. |
| **/etc** | | Contains system-wide configuration files and system databases. Originally also contained “dangerous maintenance utilities” such as init,but these have typically been moved to /sbin or elsewhere. |
| **/home** | | Contains the home directories for the users. |
| **/lib** | | Contains system libraries, and some critical files such as kernel modules or device drivers. |
| **/media** | | Default mount point for removable devices, such as USB sticks, media players, etc. |
| **/mnt** | | Stands for “mount”. Contains file system mount points. These are used, for example, if the system uses multiple hard disks or hard disk partitions. It is also often used for remote (network) file systems, CD-ROM/DVD drives, and so on. |
| **/proc** | | procfs virtual file system showing information about processes as files. |
| **/root** | | The home directory for the super user “root” – that is, the system administrator. This account’s home directory is usually on the initial file system, and hence not in /home (which may be a mount point for another filesystem) in case specific maintenance  needs to be performed, during which other filesystems are not available. Such a  case could occur, for example, if a hard disk drive suffers physical failures and cannot be  properly mounted. |
| **/tmp** | | A place for temporary files. Many systems clear this directory upon startup; it might  have tmpfs mounted atop it, in which case its contents do not survive a reboot, or it  might be explicitly cleared by a startup script at boot time. |
| **/usr** | | Originally the directory holding user home directories,its use has changed. It  now holds executables, libraries, and shared resources that are not system critical, l  ike the X Window System, KDE, Perl, etc. However, on some Unix systems, some user  accounts may still have a home directory that is a direct subdirectory of /usr, such as the  default as in Minix. (on modern systems, these user accounts are often related to  server or system use, and not directly used by a person). |
| **/usr/bin** | | This directory stores all binary programs distributed with the operating system not residing  in /bin, /sbin or (rarely) /etc. |
| **/usr/include** | | Stores the development headers used throughout the system. Header files are mostly  used by the **#include**directive in C/C++ programming language. |
| **/usr/lib** | | Stores the required libraries and data files for programs stored within /usr or else  where. |
| **/var** | | A short for “variable.” A place for files that may change often – especially in size,  for example e-mail sent to users on the system, or process-ID lock files. |
| **/var/log** | | Contains system log files. |
| **/var/mail** | | The place where all the incoming mails are stored. Users (other than root) can  access their own mail only. Often, this directory is a symbolic link to /var/spool/mail. |
| **/var/spool** | | Spool directory. Contains print jobs, mail spools and other queued tasks. |
| **/var/tmp** | | A place for temporary files which should be preserved between system reboots. |

**Unix file system:**



## ****Directories or Files and their Description****



**FilePermissions:**

EveryfileanddirectoryinyourUNIXsystemhasfollowing3permissionsdefinedfortheuser, groupandothers.

* + **Read:**This permissiongivesyoutheauthoritytoopenandreada file. Readpermissionona directorygives youthe ability to lists its content.
  + **Write:**The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
  + **Execute:**In Unix/Linux,you cannot run a program unless the execute permission is set.If the execute permissionis not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

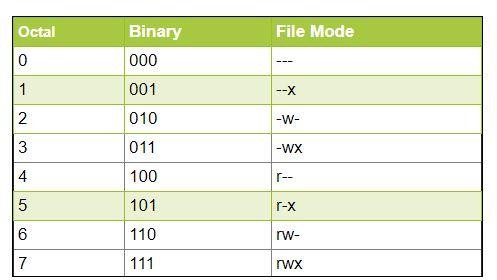
'**chmod'**commandwhichstands for 'changemode'can set permissions (read, write, execute) ona file/directoryfor the owner, group and the world.

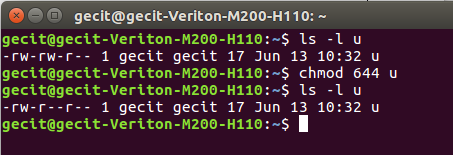
**Syntax:**chmodpermissionsfilename There are 2 ways to use the command -

1. **Absolutemode**
2. **Symbolicmode**

**Absolute(Numeric)Mode**

Inthismode,filepermissionsarenotrepresentedascharactersbutathree-digitoctalnumber.



**Output:**

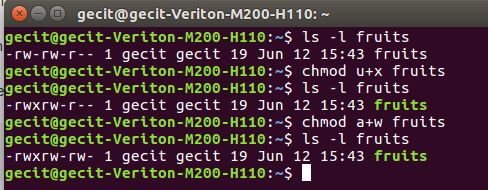
**Symbolic Mode:** In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+** | Addsapermissiontoa fileordirectory |
| **-** | Removesthepermission |
| **=** | Setsthepermissionandoverridesthe permissions set earlier. |

Thevariousownersarerepresentedas-

|  |  |
| --- | --- |
| **UserDenotations** | |
| u | user/owner |
| g | group |
| o | other |
| a | all |

**Output:**

****

4.Execution of User Management Commands like who, whoami, su, sudo, login, logout, exit, passwd, useradd/adduser, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, chown, chage, chgrp, chfn, etc

Description:

## ****1. who****

**Description**: Displays a list of users currently logged into the system.  
**Syntax**: who [options]  
**Example**:

who

Output:

user1 tty1 Jan 08 08:15

user2 pts/0 Jan 08 09:02

## ****2. whoami****

**Description**: Displays the current logged-in user's username.  
**Syntax**: whoami  
**Example**:

whoami

Output:

user1

## ****3. su****

**Description**: Switches to another user account.  
**Syntax**: su [username]  
**Example**:

su admin

After entering the password, the shell switches to the admin user.

## ****4. sudo****

**Description**: Executes commands as another user, typically root.  
**Syntax**: sudo [command]  
**Example**:

sudo apt update

Executes the apt update command with root privileges.

## ****5. login****

**Description**: Logs into a system or user account. Typically used in terminals.  
**Syntax**: login [username]  
**Example**:

login user1

Prompts for the user's password and logs into the account.

## ****6. logout****

**Description**: Logs out from the current session.  
**Syntax**: logout  
**Example**:

logout

## ****7. exit****

**Description**: Exits the current shell or session.  
**Syntax**: exit [n]  
**Example**:

exit

If you pass a numeric value n, it returns that as the exit status.

## ****8. passwd****

**Description**: Changes a user's password.  
**Syntax**: passwd [username]  
**Example**:

passwd

Prompts the user to enter a new password.

## ****9. useradd/adduser****

**Description**: Adds a new user to the system.  
**Syntax**:

* useradd [options] username
* adduser [username] (Debian-based systems)

**Example**:

useradd -m -s /bin/bash newuser

Adds a user newuser with a home directory and bash shell.

## ****10. usermod****

**Description**: Modifies an existing user's account details.  
**Syntax**: usermod [options] username  
**Example**:

usermod -l newusername oldusername

Renames oldusername to newusername.

## ****11. userdel****

**Description**: Deletes a user account.  
**Syntax**: userdel [options] username  
**Example**:

userdel -r username

Deletes the user and their home directory.

## ****12. groupadd****

**Description**: Adds a new group.  
**Syntax**: groupadd [options] groupname  
**Example**:

groupadd developers

Adds a group named developers.

## ****13. groupmod****

**Description**: Modifies an existing group.  
**Syntax**: groupmod [options] groupname  
**Example**:

groupmod -n newgroupname oldgroupname

Renames oldgroupname to newgroupname.

## ****14. groupdel****

**Description**: Deletes a group.  
**Syntax**: groupdel groupname  
**Example**:

groupdel developers

## ****15. gpasswd****

**Description**: Administers /etc/group to set or update group passwords.  
**Syntax**: gpasswd [options] groupname  
**Example**:

gpasswd developers

Prompts to set a password for the developers group.

## ****16. chown****

**Description**: Changes ownership of a file/directory.  
**Syntax**: chown [owner][:[group]] file  
**Example**:

chown user1:developers file.txt

Sets user1 as the owner and developers as the group for file.txt.

## ****17. chage****

**Description**: Changes user account expiration details.  
**Syntax**: chage [options] username  
**Example**:

chage -l username

Displays password aging details for the user.

## ****18. chgrp****

**Description**: Changes the group ownership of a file/directory.  
**Syntax**: chgrp groupname file  
**Example**:

chgrp developers file.txt

Changes the group of file.txt to developers.

## ****19. chfn****

**Description**: Changes a user's full name and other information.  
**Syntax**: chfn [options] username  
**Example**:

chfn username

Prompts the user to update personal details like full name and phone number.

### Notes:

* **Root Privileges**: Many of these commands require root or sudo privileges.
* **Best Practices**: Always confirm critical changes (like deleting users) to avoid system issues.

.

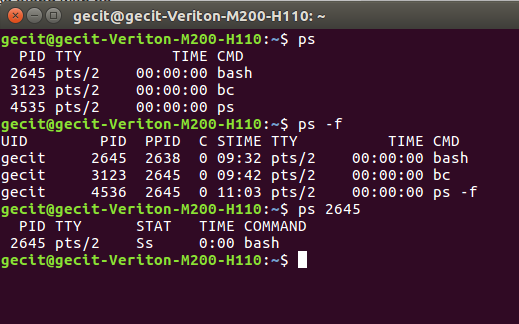
5.ToPracticeonprocess,disk,networkutilities

# Processutilities:

AninstanceofaprogramiscalledaProcess. Insimpleterms, anycommandthat yougiveto yourunixmachine starts a new process.

# ps:

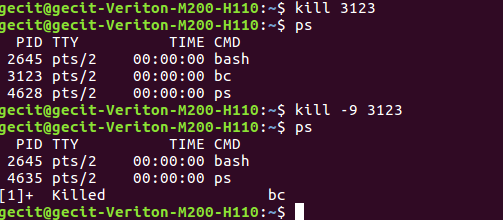
* + ps(Processstatus) canbeusedtosee/listalltherunningprocesses.
  + Formoreinformation-f(full) canbe usedalongwith ps
  + Forasingleprocess information,psalongwithprocess idis used



# kill:

Wheninvokedkillcommandsendsaterminationsignaltotheprocessbeingkilled. Wecanemploysurekill signal to forcibly terminate a process. Signal number for sure kill is 9

**$kill Pid:**

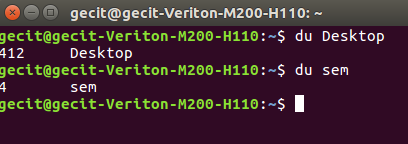
****

**DiskUtilities:**

**du:(Diskusuage)**

ducommand-lineutilityhelpsyoutofindoutthediskusageofset offilesoradirectory.

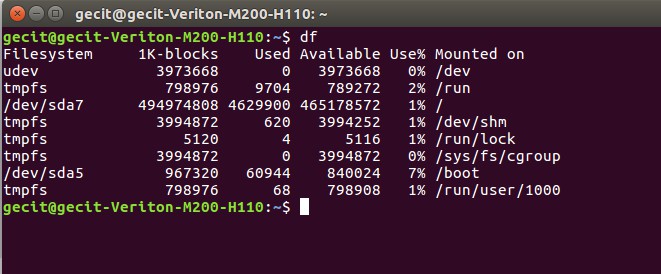
**Syntax:**du[OPTION]...[FILE]…



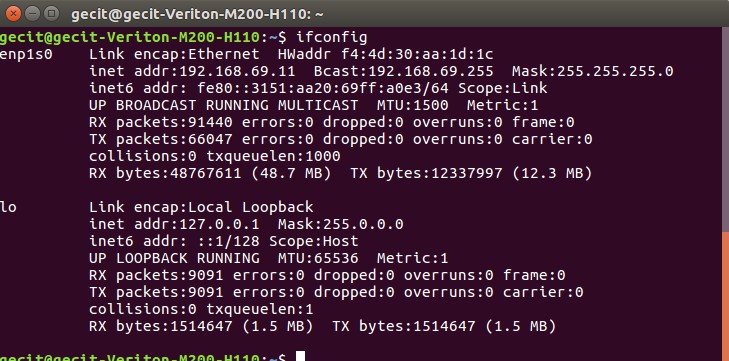
**df:(Diskfreespace)**

dfcommand displaystheamountofdiskspaceavailableonthefilesystemcontainingeachfilenameargument.

**Syntax:**df[OPTION]...[FILE]...

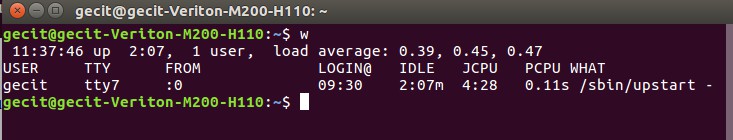


# NetworkUtilities:

1. **ifconfig:**Itisusedtoconfigurenetworkinterfaceparameters.

# w:

wprintsa summaryofthecurrentactivityonthesystem, includingwhateachuserisdoing, andtheirprocesses.



5.

6. TO PRACTICE **Memory Management Commands** in UNIX/Linux with **description**, **syntax**, and **examples**.

## ****1. free****

**Description**: Displays the total, used, and available memory (RAM and swap) on the system.  
**Syntax**: free [options]  
**Example**:

free -h

Output: Displays memory information in human-readable format.

## ****2. /proc/meminfo****

**Description**: A file that provides detailed information about memory usage from the system's perspective.  
**Syntax**: cat /proc/meminfo  
**Example**:

cat /proc/meminfo

Output: Displays details like total memory, free memory, buffers, cached memory, etc.

## ****3. top****

**Description**: Provides a dynamic real-time view of system processes, including memory and CPU usage.  
**Syntax**: top  
**Example**:

top

Press M in the top interface to sort processes by memory usage.

## ****4. htop****

**Description**: An enhanced, user-friendly version of top with an interactive UI.  
**Syntax**: htop  
**Example**:

htop

Use arrow keys to navigate and F keys for actions (e.g., killing a process).

## ****5. df****

**Description**: Reports file system disk space usage.  
**Syntax**: df [options] [file]  
**Example**:

df -h

Output: Displays disk space usage in a human-readable format for all mounted filesystems.

## ****6. du****

**Description**: Estimates and displays file or directory disk usage.  
**Syntax**: du [options] [directory/file]  
**Example**:

du -sh /var/log

Output: Shows the size of the /var/log directory in human-readable format.

## ****7. vmstat****

**Description**: Reports system performance, including memory, CPU, and IO statistics.  
**Syntax**: vmstat [options] [delay [count]]  
**Example**:

vmstat 2 5

Displays system performance statistics every 2 seconds, 5 times.

## ****8. dmidecode****

**Description**: Displays detailed hardware information, including memory details, by decoding the DMI table.  
**Syntax**: dmidecode [options]  
**Example**:

sudo dmidecode -t memory

Output: Shows detailed information about the system's memory modules.

## ****9. sar****

**Description**: Collects, reports, and saves system activity, including memory usage.  
**Syntax**: sar [options]  
**Example**:

sar -r 2 5

Reports memory usage statistics every 2 seconds for 5 iterations.

## ****10. pagesize****

**Description**: Displays the size of a memory page in bytes.  
**Syntax**: pagesize  
**Example**:

pagesize

Output: Returns the memory page size, typically 4096 bytes on most systems.

### ****Examples with Common Use Cases****

1. **Check Available Memory**:
2. free -m

Displays memory in MB.

1. **Monitor Real-Time Processes and Memory Usage**:
2. top

Press M to sort by memory.

1. **Analyze Disk Usage of a Directory**:
2. du -h --max-depth=1 /home

Lists sizes of all subdirectories in /home.

1. **Get Swap Usage with sar**:
2. sar -S 2 5

Reports swap space usage every 2 seconds for 5 iterations.

1. **View Detailed Memory Hardware Information**:
2. sudo dmidecode -t memory

### ****Tips for Memory Management****

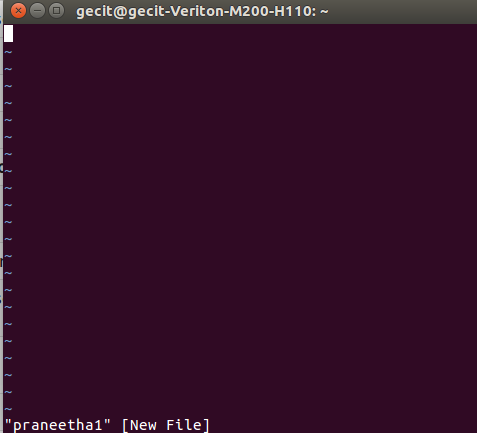
* Use htop for interactive monitoring.
* Automate system monitoring with sar or write scripts using /proc/meminfo.
* For hardware upgrades, use dmidecode to identify memory specifications.
* Regularly check and clean up unnecessary files using du and df.

7.TopracticeUNIXcommandsonViEditor.

**Vi Editor:**The default editor that comes with the UNIX operating system is called vi (visual editor). Using vi editor, wecan edit an existing file or createa new file. Wecan also use this editor to just read a text file.

**Syntax:**vifilename

$vipraneetha1



ToworkonVIeditor,youneedtounderstand itsoperationmodes.Theycanbedividedintotwomainparts.

1. Commandmode:

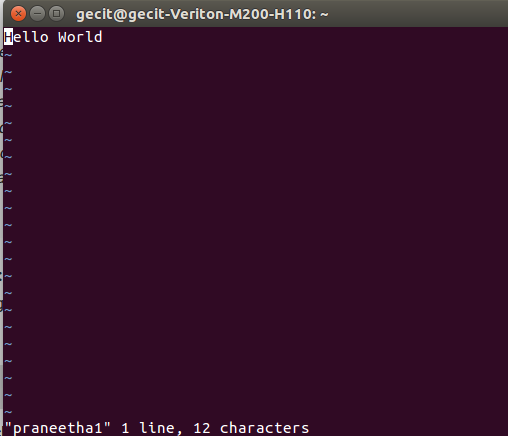
Thevieditor opensinthismode,anditonlyunderstands commands

* + Inthismode, youcan,movethecursorandcut, copy,pastethetext
  + Thismodealsosavesthechangesyouhavemadetothefile
  + Commandsarecasesensitive.Youshouldusetheright lettercase.

1. Insertmode:
   * Thismodeisfor insertingtextinthefile.
   * YoucanswitchtotheInsertmodefromthecommandmode bypressing'i' onthekeyboard
   * OnceyouareinInsert mode,anykey wouldbetakenasaninput forthefileonwhichyouarecurrently working.
   * ToreturntothecommandmodeandsavethechangesyouhavemadeyouneedtopresstheEsckey

**viEditingcommands:**

**Note**:Youshould be inthe"command mode"to executethesecommands.VIeditoriscase-sensitiveso make sure you type the commands in the right letter-case.



ToExitvi:

|  |  |  |
| --- | --- | --- |
|  | :x<Return> | *quitvi,writingout modifiedfiletofilenamed in original invocation* |
|  | :wq<Return> | *quitvi,writingout modifiedfiletofilenamedinoriginal invocation* |
|  | :q<Return> | *quit (orexit) vi* |

InsertingorAddingText:

|  |  |  |
| --- | --- | --- |
|  | i | *Inserttextbeforecursor, until<Esc>hit* |
|  | I | *inserttextatbeginningofcurrentline, until <Esc>hit* |
|  | a | *appendtextaftercursor,until<Esc>hit* |
|  | A | *appendtextto endofcurrent line,until<Esc>hit* |
|  | o | *openandputtextinanewlinebelowcurrentline,until<Esc>hit* |
|  | O | *openandputtextinanewline above currentline,until<Esc>hit* |

ChangingText:

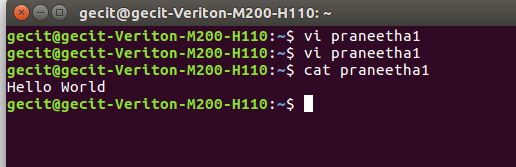
|  |  |  |
| --- | --- | --- |
|  | r | *replacesinglecharacterundercursor(no<Esc>needed)* |
|  | R | *replacecharacters, startingwithcurrentcursorposition, until <Esc>hit* |
|  | cw | *changethecurrentwordwithnewtext,*  *startingwiththecharacterundercursor, until<Esc>hit* |
|  | C | *change(replace) thecharactersinthecurrentline,until<Esc>hit* |
|  | cc | *change(replace)theentirecurrentline, stoppingwhen<Esc>ishit* |

DeletingText:

|  |  |  |
| --- | --- | --- |
|  | x | *deletesinglecharacterunder cursor* |

|  |  |  |
| --- | --- | --- |
|  | dw | *deletethesinglewordbeginningwithcharacterundercursor* |
|  | D | *deletetheremainderoftheline,startingwithcurrentcursorposition* |
|  | dd | *irecurrentline* |

**Output:**

****

Practiceremainingcommandsinexperiment1onvieditor

6.

6

6.

8.

a) Study of Shell, Types of Shell, Variables andOperators

b) Execute the following Scripts (at least 6):

(i) Write a shell script to perform arithmetic operations.

(ii) Write a shell script to calculate simple interest.

(iii) Write a shell script to determine largest among three integer numbers.

(iv) Write a shell script to determine a given year is leap year or not

(v) Write a shell script to print multiplication table of given number using while statement.

(vi) Write a shell script to search whether element is present is in the list or not.

(vii) Write a shell script to compare two strings.

(viii) Write a shell script to read and check if the directory / file exists or not, if not make the directory / file.

(ix) Write a shell script to implement menu-driven calculator using case statement.

(x) Write a shell script to print following pattern:

\*

\* \*

\* \* \*

\* \* \* \*

(xi) Write a shell script to perform operations on directory like: display name of current directory; display list of directory contents; create another directory, write contents on that and copy it to a suitable location in your home directory; etc.

7. Execute the following scripts using grep / sed commands:

(i) Write a script using grep command to find the number of words character, words and lines in a file.

(ii) Write ascriptusing egrep command to display list of specific type of files in the directory.

(iii) Write a script using sed command to replace all occurrences of particular word in given a file.

(iv) Write a script using sedcommand to print duplicated lines in input.

8.

Execute the following scripts using awk / perl languages:

(i) Write an awk script to print all even numbers in a given range.

(ii) Write an awk script to develop a Fibonacci series (take user input for number of terms).

(iii) Write a perl script to sort elements of an array.

(iv) Write a perl script to check a number is prime or no

**Aim:**Toprintthefactorialoffirstnnaturalnumbers.

**Program:**

echo"enterrange" read n

i=1

while[$i-le$n] do

j=1 fact=1

while[$j-le$i] do

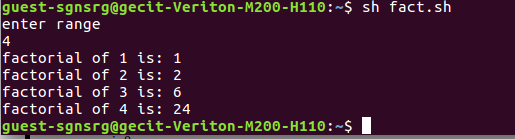
fact=`expr$fact\\*$j` j=`expr $j + 1`

done

echo factorialof$iis:$fact i=`expr $i + 1`

done

**Output:**

****

1. Writeashellscriptto generateamultiplicationtableofthegivennumber. **Aim:**To generate a multiplication table of the given number. **Description:**

To display any information on the screen we can use ‘echo‘ command . The text within the double quoteswas displayed on the screen. ‘expr’ command is used to evaluate the expression. The expression within the symbol ‘ ‘ is evaluated byusing the expr command $variable gives the value ofthe variable it contained expr command combines two functions perfumes arithmetic operations on integers manipulate strings

Theoperand+,-,\*etc..,mustbe enclosedoneither sidebywhite space.

**Program:**

echo"whichnumbertogeneratemultiplicationtable" read number

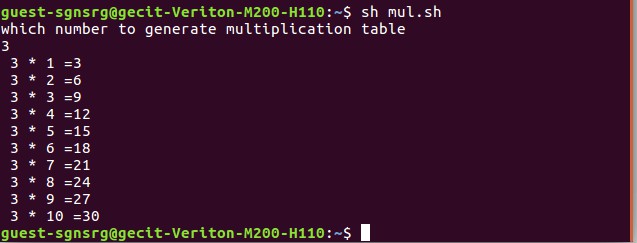
i=1

while[$i-le10] do

echo "$number\*$i=`expr$number \\*$i`" i=`expr $i + 1`

done

**Output:**

****

1. Writeashellscript thatcountsthenumberoflinesandwordspresentinagiven file.

**Aim:**Tocountsthenumber oflinesandwordspresentinagiven file.

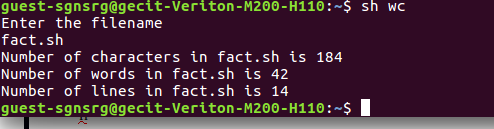
**Description:**wccommand isusedto displaythenumberofcharacters, words,lines inagivenfile. “|” inputs the output of file content to wc.

**Program:**

echoEnterthefilename read file

w=`cat$file|wc-w` c=`cat $file | wc -c` l=`cat $file | wc -l`

echoNumberofcharactersin$fileis$c echo Number of words in $file is $w echo Number of lines in $file is $l



1. Writeashellscriptthatdisplaysthelistofallfiles inthegivendirectory.

**Aim:**Todisplaythelistofallfilesinthegiven directory

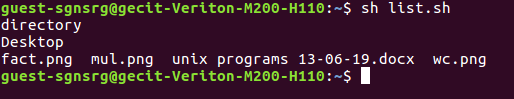
**Program:**

echodirectory read d

forfilein$d do

if[-d$file] then

ls$file fidone



1. Write a shell script (small calculator)that adds, subtracts, multiplies and divides the given two integers. Therearetwodivisionoptions:onereturnsthequotient andtheotherreturnsreminder.Thescriptrequires3 arguments: The operation to be used and two integer numbers. The options are add (-a), subtract (-s), multiply (-m), quotient (-c) and reminder (-r).

# Program:

echo"enterfirstvalue" read x

echo"EnterSecondValue" read y

while[${q:-1}-ne0] do

echo "Enter -a for adding" echo"Enter-sforsubtraction"

echo"Enter-mformultiplication" echo "Enter -c for Quotient"

echo"Enter-rforreminder" echo choice

read b case$bin

-a)p=`expr$x+$y` echo "Sum = $p"

;;

-s)p=`expr $x - $y` echo"difference=$p"

;;

-m)p=`expr$x\\*$y` echo "Product = $p"

;;

-c)p=`expr $x/ $y` echo "quotient=$p"

;;

-r)p=`expr$x%$y` echo "reminder=$p"

;;

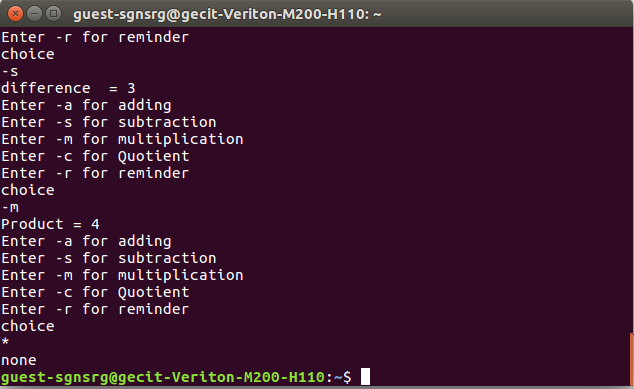
\*)echo"none"break

;;

esac done

# Output:

****

****

1. Writeaprogramthattakesoneormore file/directorynamesascommand line input andreportsthefollowing information on the file: File type, Number of links, Time of last access, Read, Write and Execute permissions.

# Program:

#include<stdio.h> #include<unistd.h> #include<dirent.h> #include<fcntl.h> #include<sys/stat.h> #include<sys/types.h> #include<time.h>

intmain(intargc,char \*argv[])

{

char \*at,\*mt,\*ct; struct stat buf; stat(argv[1],&buf);

printf("theinodenois:%ld\n",buf.st\_ino); if(S\_ISDIR(buf.st\_mode))

printf("it is a directory"); if(S\_ISREG(buf.st\_mode)) printf("itisregurlarfile\n");

printf("thenooflinksis%ld\n",buf.st\_nlink); at=ctime(&buf.st\_atime);

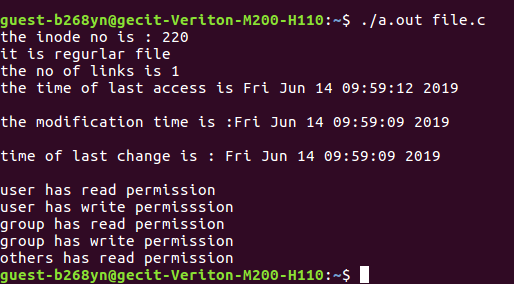
printf("thetimeoflastaccessis%s\n",at); mt=ctime(&buf.st\_mtime);

printf("themodificationtimeis:%s\n",mt); ct=ctime(&buf.st\_ctime);

printf("time of last change is : %s\n",ct); if((buf.st\_mode &S\_IRUSR)==S\_IRUSR) printf("user has read permission\n"); if((buf.st\_mode &S\_IWUSR)==S\_IWUSR) printf("user has write permisssion\n"); if((buf.st\_mode &S\_IXUSR)==S\_IXUSR) printf("user has execute permission\n"); if((buf.st\_mode &S\_IRGRP)==S\_IRGRP) printf("group has read permission\n"); if((buf.st\_mode &S\_IWGRP)==S\_IWGRP) printf("group has write permission\n"); if((buf.st\_mode &S\_IXGRP)==S\_IXGRP) printf("group has execute permission\n"); if((buf.st\_mode &S\_IROTH)==S\_IROTH) printf("others has read permission\n"); if((buf.st\_mode&S\_IWOTH)==S\_IWOTH) printf("others has write permission\n"); if((buf.st\_mode &S\_IXOTH)==S\_IXOTH) printf("others has execute permisssion\n");

}

**Output:**

****

1. Write a C program that illustrates uses of the opendir, readdir, and closedir APIs.

# Program:

#include<stdio.h> #include<fcntl.h> #include<dirent.h> #include<unistd.h>

int main()

{

chard[10]; DIR \*e;

structdirent\*sd;

printf("enterdirnametoopen:"); scanf("%s",d);

e=opendir(d); if(e==NULL)

printf("dirdoesnotexist"); else

{

printf("dir exist\n"); while((sd=readdir(e))!=NULL)

{

printf("%s\t",sd->d\_name);

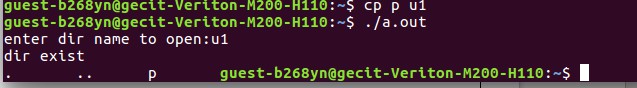
}

closedir(e);

}

}

# Output:

****

1. WriteaCprogramthatcountsthenumber ofblanksinatextfileusingstandard I/O

# Program:

#include <fcntl.h> #include<sys/stat.h> #include <stdio.h>

intmain(intargc,char\*\*argv)

{

FILE \*fd1;intn,count=0; char buf;

fd1=fopen(argv[1],"r"); while(!feof(fd1))

{

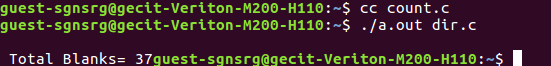
buf=fgetc(fd1); if(buf=='') count=count+1;

}

printf("\nTotalBlanks=%d",count); return (0);

}

Output:



1. Implement inC, Unixcommand‘cat’usingsystemcalls.

# Program:

#include<fcntl.h> #include<sys/stat.h> #include<stdio.h> #include<unistd.h>#define BUFSIZE 1

intmain(intargc,char\*\*argv)

{

int fd1; int n; charbuf;

fd1=open(argv[1],O\_RDONLY); printf("displays filecontents\n"); while((n=read(fd1,&buf,1))>0)

{

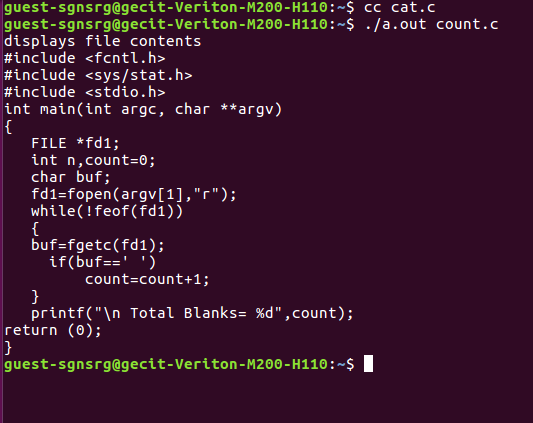
write(1,&buf,1);

}

return(0);

}

# Output:

****

1. WriteaCprogramthatillustratesthecreationofchild processusingfork systemcall.

# Program:

#include<stdio.h> #include<stdlib.h> #include<sys/types.h> #include<unistd.h>

int main(int argc,char \*argv[])

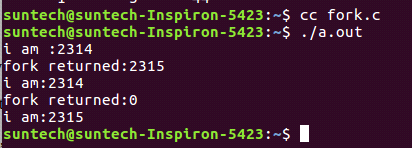
{

printf("iam:%d\n",(int)getpid()); pid\_t pid=fork();

printf("forkreturned:%d\n",(int)pid); printf("i am:%d\n",(int)getpid()); return 0;

}

# output:

****

1. WriteaCprogramthatillustrateshowtoexecutetwocommandsconcurrentlywitha command pipe.

# Program:

#include<stdio.h> #include<fcntl.h> #include<unistd.h> #include<sys/types.h>int main()

{

int pfd[2],p; pipe(pfd); p=fork(); if(p==0)//forchild

{

close(pfd[0]); close(1); dup(pfd[1]);

execlp("who","who", (char \*)0);

}

else

{

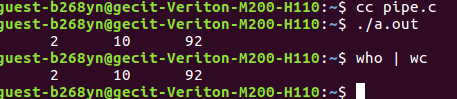
close(pfd[1]); close(0); dup(pfd[0]);

execlp("wc","wc",(char \*)0);

}

}

# output:

****

1. WriteaCprogramtoimplementtheKillfunction.

# Program:

#include<signal.h> #include<stdio.h> #include<unistd.h>void xyz(int n)

{

printf("hello:%d\n",getpid()); signal(2,xyz);

}

voidpqr(intn)

{

printf("frompqr:%d\n",getpid());

}

int main()

{

signal(2,xyz); if(fork()==0)

{

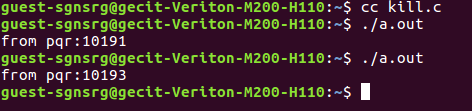
signal(2,pqr); kill(0,2);

}

while(1) return0;

}

# Output:

****

1. WriteaCprogramtoimplementtheraisefunction.

# Program:

#include <stdio.h> #include<stdlib.h> #include<signal.h>

voidsignal\_handler(intsignal)

{

/\*Displayamessageindicatingwehavereceivedasignal\*/ if (signal == SIGUSR1)

printf("ReceivedaSIGUSR1 signal\n");

/\*Exittheapplication\*/ exit(0);

}

intmain(int argc,constchar\*argv[])

{

/\*Displayamessageindicatingweareregisteringthesignalhandler\*/ printf("Registering the signal handler\n");

/\* Register the signal handler \*/ signal(SIGUSR1,signal\_handler);

/\*Displayamessageindicatingweareraisingasignal\*/ printf("Raising a SIGUSR1 signal\n");

/\*RaisetheSIGUSR1signal\*/ raise(SIGUSR1);

/\*Displayamessageindicatingweareleavingmain\*/ printf("Finished main\n");

return0;

}

# Output:

1. Writea Cprogramthatdisplaysthe realtimeofa dayevery60seconds.

**Program:**

#include<stdio.h> #include<sys/time.h> #include<sys/signal.h> #include<unistd.h>

/\*Declarations\*/ void main();

inttimes\_up();

voidmain()

{

for (;;)

{

times\_up(1);

sleep(60);

}

}

inttimes\_up(sig) int sig;

{

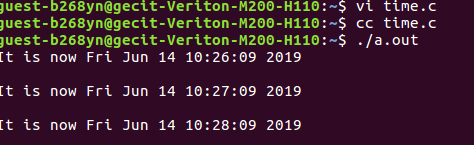
long now; structtms\*ptr; long time();

char\*ctime(); time(&now);

printf("Itisnow%s\n",ctime(&now)); return (sig);

}

# Output:

****

**AdditionalPrograms:**

1. Writeashellscripttodisplayfileswhichhasread,writeandexecutepermissions.

**Program:**

echo“enterafilename” readf

forfin`ls` do

if[-w$f–a-r $f–a–x$f] then

echo$f fi

done

1. Writeashellscripttodisplaythebiggestnumberingiveninputnumbers.

**Program:**

echo“enteranum” readnum

if[$1–ge$2] then

if[$1–ge$3] then

echo$1“isbiggestnum” else

echo$3“isbiggest num”

fi

elif[$2–ge$3] then

echo$2“isbiggernum” else

echo“Biggest numis“$3

fi

1. Write aC program that illustrates the creation of child process using fork system call. One process finds sumof even series and other process finds sum of odd series.

**Program:**

#include <stdio.h> #include<sys/types.h> #include <unistd.h> #include <fcntl.h>

intmain()

{

int i,n,sum=0; pid\_t pid; system(“clear”);

printf(“Enternvalue:”); scanf(“%d”,&n) pid=fork();

if(pid==0)

{

printf(“Fromchildprocess\n”); for(i=1;i<n;i+=2)

{

printf(“%d\”,i);

sum+=i;

}

printf(“Oddsum:%d\n”,sum);

}

else

{

printf(“Fromprocess\n”); for(i=0;i<n;i+=2)

{

printf(“%d\”,i); sum+=i;

}

printf(“Evensum:%d\n”,sum);

}

}