LINUX

Linux is an open source operating system (OS). An operating system is the software that directly manages a system’s hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all your software and the physical resources that do the work.

- Linux is an open-source operating system kernel first developed by Linus Torvalds in 1991.
- It's based on Unix and is known for its stability, security, and flexibility.
- Linux distributions (distros) package the Linux kernel with software and tools to create complete operating systems.

Users

In Linux, there are typically three types of users:

1. **Root User (Superuser)**:
   - The root user, also known as the superuser, has unrestricted access to all files and commands on the system. This user has the highest level of privileges and can perform any administrative tasks, including modifying system configurations, installing software, and managing other users.

   ```
   sudo su
   ```
2. **System Users**:
   - System users are created to run specific system processes or services. They typically have limited privileges and are used to ensure security and separation of duties on the system. System users usually do not have login shells or home directories.
3. **Regular Users**:
   - Regular users are standard users who interact with the system for everyday tasks. They have restricted access compared to the root user and are confined to their own home directories and specific permissions set by the system administrator. Regular users cannot perform administrative tasks unless granted permission via tools like sudo.

File System (support): In Linux, a file system is a method or structure used to organize and store data on a storage device, such as a hard disk drive (HDD), solid-state drive (SSD), or other storage media. It provides a way to store, retrieve, and manage files and directories within a computer system. The file system defines how data is organized, accessed, and manipulated on the storage device.

Linux supports various file systems, each with its own characteristics and features. Some of the commonly used file systems in Linux include:

1. **Ext4 (Fourth Extended File System)**:
    - Ext4 is the default file system for most Linux distributions. It is an evolution of the earlier Ext3 file system and offers improvements in performance, scalability, and reliability. Ext4 supports larger file sizes and volumes compared to Ext3.
2. **Ext3 (Third Extended File System)**:
    - Ext3 is the predecessor of Ext4 and is still widely used in many Linux distributions. It is a journaled file system, which means it maintains a journal to track changes before they are committed to the main file system, enhancing data integrity and recovery.
3. **Btrfs (B-Tree File System)**:
    - Btrfs is a modern file system designed for Linux, emphasizing features such as fault tolerance, scalability, and easy administration. It supports features like snapshots, checksums, compression, and RAID-like functionality.
4. **XFS (XFS File System)**:
    - XFS is a high-performance file system known for its scalability and reliability, particularly suitable for large-scale storage systems and high-throughput workloads. It supports features like journaling, extended attributes, and online resizing.
5. **FAT (File Allocation Table)**:
    - FAT is a simple and widely supported file system originally developed for MS-DOS. It is commonly used for external storage devices like USB flash drives and SD cards due to its cross-platform compatibility.

6. **NTFS (New Technology File System)**:
    - NTFS is a proprietary file system developed by Microsoft for the Windows operating system. While not natively supported in Linux, there are drivers and utilities available to enable read and write access to NTFS partitions.

7. **ISO 9660 (CD-ROM File System)**:

- ISO 9660 is a standard file system used for CD-ROM and DVD-ROM discs. It provides a way to organize and access data stored on optical discs in a platform-independent manner.

These are just a few examples of file systems supported by Linux. The choice of file system depends on factors such as performance requirements, scalability, data integrity, and compatibility with specific applications or hardware platforms.

File System Hierarchy:

- Understanding the Linux File System Hierarchy Standard (FHS) and the purpose of key directories like `/bin`, `/etc`, `/home`, `/var`, `/usr`, and `/tmp`

What is full form of Linux?

The full form of LINUX is Lovable Intellect Not Using XP. Linux was built by and named after Linus Torvalds. Linux is an open-source operating system for servers, computers, mainframes, mobile systems, and embedded systems.

What is the most powerful OS ?

Linux is considered to be the most efficient and remarkable operating system for software development.

Advantages of using Linux:

1. Open source. The prime advantage of using Linux is that beginners can test it out for free as it is open source.
2. Linux offers a greater degree of security than many operating systems and requires no antivirus programs for protection.
3. Linux also offers a high degree of stability, requires little disk space, has powerful networking capabilities, and puts software updates in the hands of the user.

   (Or)

   Linux offers numerous advantages, making it a popular choice for a wide range of users and organizations. Some of the key advantages of Linux include:

   Open Source and Free:
   Linux is distributed under open-source licenses, allowing users to access, modify, and redistribute the source code freely. This makes Linux distributions

(distros) and software packages available at no cost, providing significant cost savings compared to proprietary operating systems.

Customizability and Flexibility:
Linux is highly customizable, allowing users to tailor their operating system to their specific needs and preferences. Users can choose from a wide variety of Linux distributions, desktop environments, window managers, and software packages, enabling them to create a personalized computing environment.

Stability and Reliability:
Linux is known for its stability and reliability, particularly in server environments. Linux-based servers often have long uptimes and require minimal maintenance compared to other operating systems. The separation of user space and kernel space, along with robust memory management and process isolation, contribute to Linux's stability.

Security:
Linux is inherently more secure than some other operating systems due to its strong focus on security features and principles. These include built-in security mechanisms like file permissions, user authentication, access control lists (ACLs), and mandatory access controls (MAC). Additionally, the open-source nature of Linux allows for rapid identification and patching of security vulnerabilities by the community.

Performance:
Linux is known for its excellent performance and efficiency, particularly in resource-constrained environments. Linux-based systems can run on a wide range of hardware platforms, from embedded devices and smartphones to servers and supercomputers. The lightweight nature of many Linux distributions ensures optimal resource utilization and responsiveness.

Wide Range of Software:
Linux offers a vast ecosystem of open-source software and applications, including productivity tools, development environments, multimedia software, and server applications. Many popular software packages are available for Linux, and users can also choose from thousands of free and open-source software (FOSS) projects.

Scalability:
Linux scales well from small embedded systems to large-scale enterprise deployments. Its modular architecture, support for multi-core processors, and scalability features like clustering, load balancing, and virtualization make Linux suitable for a wide range of computing tasks and workloads.

Community Support:

Linux has a vibrant and active community of users, developers, and contributors worldwide. This community provides extensive documentation, tutorials, forums, mailing lists, and online resources for users seeking assistance, troubleshooting, or collaboration.

Overall, the advantages of Linux make it an attractive choice for individuals, businesses, educational institutions, and government organizations looking for a powerful, customizable, and cost-effective operating system.

Disadvantages of Linux:

However, Linux does come with certain disadvantages, including a learning curve, software and hardware compatibility challenges, and fragmentation within the ecosystem.

(or)

While Linux is a powerful and versatile operating system with many advantages, it also has some disadvantages:

Hardware Compatibility:

Linux may have limited support for certain hardware devices, particularly newer or specialized hardware components. Some hardware manufacturers may not provide Linux drivers or support, leading to compatibility issues.

Software Compatibility:

Although the availability of software for Linux has increased significantly over the years, there are still some applications and games that are not natively supported on Linux. Users may need to rely on compatibility layers like Wine or virtualization to run certain Windows applications.

User Interface Consistency:

Linux distributions often come with a variety of desktop environments and window managers, leading to differences in user interface design and behavior. This lack of consistency can be confusing for new users switching between distributions or desktop environments.

Fragmentation:

The open-source nature of Linux has led to the proliferation of numerous distributions (distros), each with its own package management system, software repositories, and configuration tools. This fragmentation can make it challenging for users to choose the right distribution and maintain compatibility across different systems.

Learning Curve:

Linux can have a steep learning curve for users accustomed to other operating systems like Windows or macOS, particularly when it comes to command-line usage, system configuration, and troubleshooting. New users may need to invest time in learning Linux concepts and commands.

Limited Commercial Support:

While there are many resources available for community support, Linux may lack the same level of commercial support and enterprise-grade solutions offered by proprietary operating systems like Windows or macOS. This can be a concern for businesses requiring dedicated support contracts and service-level agreements.

Gaming Support:

Although gaming support on Linux has improved in recent years with platforms like Steam and Proton, the selection of native Linux games is still relatively limited compared to Windows. Users may encounter compatibility issues or performance limitations when running certain games on Linux.

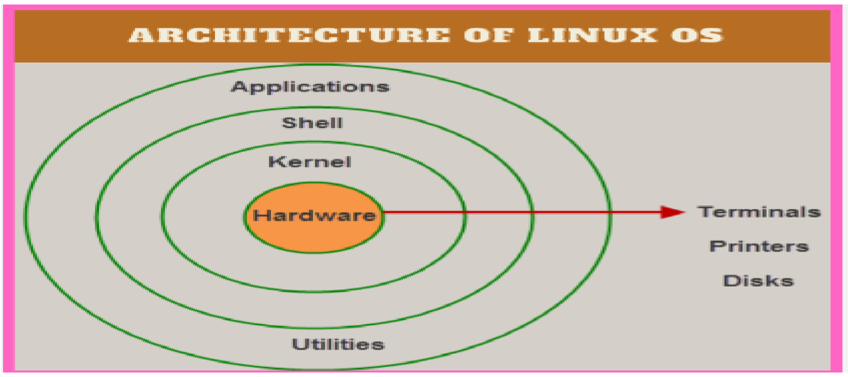
Peripheral Support:

While Linux has broad support for many peripherals and devices, there may be compatibility issues with certain printers, scanners, webcams, and other peripherals, particularly those designed for Windows or macOS.

what is linux architecture:

The architecture of Linux is the underlying structured layer like other operating systems. Generally, it has four fundamental layers.

Those are: application, shell, kernel, and hardware. It is very important to understand for both the developers and users how each layer is connected and creates the whole system.



Applications:

The topmost layer of the Linux architecture consists of applications. These are the software programs that you, as the user, interact with directly. They range from system applications like file managers, text editors, and network managers, to user applications like browsers.

While applications communicate with the hardware through the kernel, they interact with the user through the shell. For instance, when you run a command to open a file in a text editor, the shell interprets your command, the kernel fetches the file from the hardware, and the text editor (application) displays it.

User Space: Above the kernel is the user space, which includes all non-kernel processes and user-level applications running on the system. User space encompasses a wide range of software components, including system utilities, graphical user interfaces (GUIs), desktop environments, command-line interfaces (CLIs), and user applications.

What about GUIs & CLIs :

Graphical User Interface (GUI) Applications:

Graphical user interface (GUI) applications provide users with visual interfaces for interacting with the operating system and performing tasks. Examples include web browsers, office suites, media players, graphics editors, and games. GUI applications typically use window managers, desktop environments, and display servers to render graphical elements on the screen.

Command-Line Interface (CLI) Applications:

Command-line interface (CLI) applications are text-based programs that allow users to interact with the operating system using command-line commands and scripts. These applications are executed in terminal emulators and provide powerful tools for system administration, automation, scripting, and programming. Examples include shell utilities, text editors, version control systems, package managers, and compilers.

System Utilities and Services:

Linux includes a wide range of system utilities and services that perform essential tasks such as system configuration, administration, networking, security, logging, and monitoring. These utilities and services are often command-line based but may also include GUI-based tools for user convenience. Examples include systemd, NetworkManager, syslog-ng, cron, and various networking utilities.

Server Applications:

Linux is widely used as a server platform, hosting a multitude of server applications and services for web hosting, file sharing, email, databases, virtualization, containerization, cloud computing, and more. Server applications typically run in the background as daemons or services, providing networked services to clients or other systems.

Development Tools and Environments:

Linux provides comprehensive development tools and environments for software development, including compilers, interpreters, libraries, debuggers, integrated development environments (IDEs), text editors, version control systems, and build systems. These tools enable developers to create, debug, and deploy software applications for Linux and other platforms.

Shell:

The shell is a command-line interpreter that allows users to interact with the operating system and execute commands. Linux supports various shell implementations, including Bash (Bourne Again Shell), which is the default shell for most Linux distributions. Other popular shells include Zsh (Z Shell), Fish (Friendly Interactive Shell), and Ksh (Korn Shell).

Kernel:

At the core of Linux is the Linux kernel, which serves as the foundation of the operating system. The kernel is responsible for managing hardware resources, providing essential system services, and facilitating communication between software and hardware. It handles tasks such as process management, memory management, device drivers, file system management, and system calls.

Hardware:

The lowest level of the Linux architecture is the hardware layer. This layer comprises the physical components of a computer, such as the hard drive, RAM, motherboard, CPU, network interfaces, and peripherals. These components are the tangible pieces of your system on which the rest of the architecture is built.

These hardware components include:

Central Processing Unit (CPU):

The CPU is the primary processing unit of a computer, responsible for executing instructions and performing calculations. The Linux kernel interacts with the CPU to schedule processes, manage system resources, and execute system calls.

Memory (RAM):

Memory, or Random Access Memory (RAM), is used by the CPU to store data and program instructions temporarily during program execution. The Linux kernel manages memory allocation, paging, and virtual memory to ensure efficient use of available memory resources.

Storage Devices:

Storage devices, such as hard disk drives (HDDs), solid-state drives (SSDs), and optical drives, are used to store data and files permanently. The Linux kernel includes drivers for various storage devices and file systems, allowing users to access and manage data stored on these devices.

Input/Output (I/O) Devices:

Input/Output (I/O) devices enable communication between the computer system and external devices, such as keyboards, mice, monitors, printers, network adapters, and USB devices. The Linux kernel includes drivers for a wide range of I/O devices, allowing users to interact with and control these devices.

Network Interface Cards (NICs):

Network Interface Cards (NICs) enable connectivity to networks, allowing the computer system to communicate with other devices and access remote resources. The Linux kernel includes network drivers for various NICs, supporting wired and wireless networking technologies.

Graphics Processing Unit (GPU):

The Graphics Processing Unit (GPU) is responsible for rendering graphics and images on the computer screen. Linux supports a wide range of GPUs from different manufacturers, with drivers provided by the GPU vendors or open-source community.

Motherboard and Chipset:

The motherboard and chipset provide the physical infrastructure and connectivity for other hardware components to communicate with each other. The Linux kernel includes drivers for motherboard components and chipsets, ensuring compatibility and functionality.

Firmware and BIOS/UEFI:

Firmware, including Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI), provides low-level control and initialization of hardware components during the boot process. The Linux kernel interacts with firmware to initialize hardware devices and establish the initial system configuration.

--> In the Linux ecosystem, utilities refer to command-line tools, programs, and applications designed to perform specific tasks or functions. These utilities are essential for system administration, automation, troubleshooting, and everyday use. Here are some categories of utilities commonly found in Linux:

File Management

ls: List directory contents

```
ubuntu@ip-172-31-32-223:~$ ls
file1  file2  project  workspace
```

touch: Creates an empty file or updates the timestamp of an existing file.

```
ubuntu@ip-172-31-32-223:~$ touch file4 (To create an empty file)
ubuntu@ip-172-31-32-223:~$ ls
file1 file2  file3  file4
```

*For example, have to create a 100 files at a time

```
ubuntu@ip-172-31-32-223:~$ touch file{4..100}
```

Directory Management

cd: Changes the current directory

```
ubuntu@ip-172-31-32-223:~$ ls
file1  file2  project
ubuntu@ip-172-31-32-223:~$ cd project
ubuntu@ip-172-31-32-223:~/project$
```

pwd: Prints the current working directory.

```
ubuntu@ip-172-31-32-223:~$ pwd
/home/ubuntu
```

mkdir: Creates a new directory.

```
file1  file2  file3  project
ubuntu@ip-172-31-32-223:~$ mkdir Media
ubuntu@ip-172-31-32-223:~$ ls
Media file1  file2  file3 Project
```

rm: Removes files or directories.

```
ubuntu@ip-172-31-32-223:~$ ls
file1  file2  file3  project
ubuntu@ip-172-31-32-223:~$ rm file3  (If file3 is complete empty)
ubuntu@ip-172-31-32-223:~$ rm -rf file3 (If file3 has some data)
ubuntu@ip-172-31-32-223:~$ ls
file1  file2  project
```

rmdir: Removes empty directories.

```
ubuntu@ip-172-31-32-223:~$ ls
Media file1  file2  Project
ubuntu@ip-172-31-32-223:~$ rmdir Media
ubuntu@ip-172-31-32-223:~$ ls
```

file1 file2 Project

File and Directory Manipulation:

cp: Copies files or directories.

```
ubuntu@ip-172-31-32-223: ~$ cp file2 file3
```

(The entire data of file2 will be copy in file3 & new file {file3} created itself by giving these command)

mv: Moves or renames files or directories.

The primary use of mv command is to move and rename files and directories.

```
ubuntu@ip-172-31-32-223: ~$ mv file2 file3
```

The data of file2 will be stored in file3 & complete file2 will be deleted.

Text Display

cat: Concatenates and displays files.

```
ubuntu@ip-172-31-32-223:~$ sudo cat  >file1 (To create file & enter content)
Hi I'm Prasanth Reddy
ubuntu@ip-172-31-32-223:~$ cat file2 (To display content)
Hi how are you.
Where are you working?
ubuntu@ip-172-31-32-223:~$sudo cat file1 file2>file3
```

(Both the contents of file1 & file2 will be merge in file3)

o/p: Hi I'm Prasanth Reddy

Hi how are you.

Where are you working?

more: Displays file contents one screen at a time.

```
ubuntu@ip-172-31-32-223:~$ more file2
```

1.Hi how are you.

2.Where are you working?

…………………………

………………………….

60. Working in CtrlAltFix Solutions

For example, here assume that there is 60 lines but while giving more command it will display approximately 15 to 20 Lines at a time on a screen page. But we must see next lines we have to click on space bar to see next page up to reached end.

After reached to end of page press q to exit.

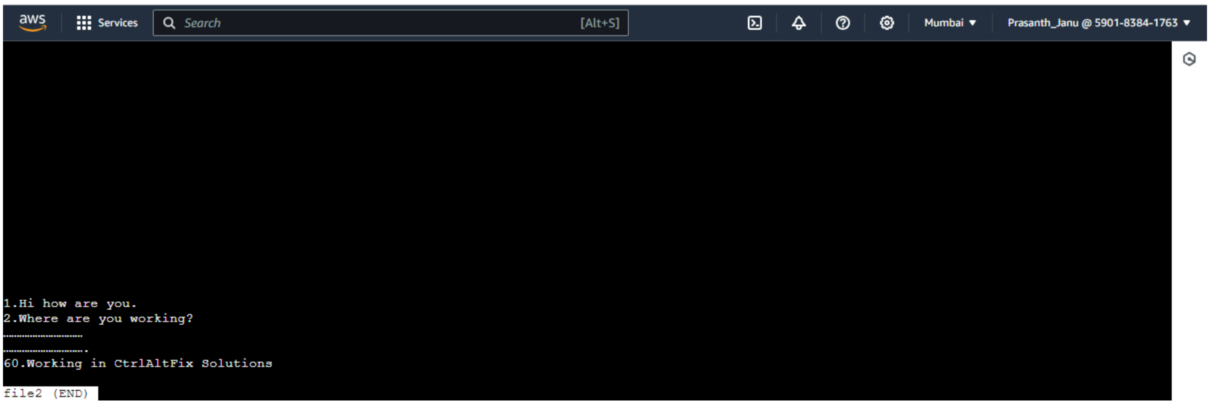
To view each number, click on enter tab.

***** Note: What is use of more command

The main use of more command is displaying the whole content in page by page.

less: Displays file contents with backward movement.

ubuntu@ip-172-31-32-223:~$ less file2



```
1.Hi how are you.
2.Where are you working?
……………………
…………………….
60.Working in CtrlAltFix Solutions
file2 (END)
```

Same as more command both are similar.

head: Displays the beginning of a file.

ubuntu@ip-172-31-32-223:~$ sudo head file2

```
1.Hi how are you.
2.Where are you working?
…………………………
………………………….
60.Working in CtrlAltFix Solutions
ubuntu@ip-172-31-32-223:~$ sudo head -n3 file2
1.Hi how are you.
2.Where are you working?
…………………………
```

(or)

```
ubuntu@ip-172-31-32-223:~$ sudo head -n -3 file2
1.Hi how are you.
2.Where are you working?
…………………………
```

tail: Displays the end of a file.

```
ubuntu@ip-172-31-32-223:~$ sudo tail file2
1.Hi how are you.
2.Where are you working?
…………………………
………………………… .
60.Working in CtrlAltFix Solutions
--------------space--------------------------
{
ubuntu@ip-172-31-32-223:~$ sudo tail -n3  file2
………………………… .
```

```
60.Working in CtrlAltFIX Solutions
---------------space-------------
```

→This command will display the data from line where we give to end.

```
ubuntu@ip-172-31-32-223:~$ sudo tail -n +4 file2
………………………… .
60.Working in CtrlAltFIX Solutions
----------------------------------------
```

Text Search and Filtering:

grep: Searches for patterns in files.

grep is a powerful command-line utility for searching plain-text data sets for lines that match a regular expression pattern.

-
  ```
  ubuntu@ip-172-31-32-223:~$ grep "Prasanth" file3
  ```

```
Hi this is Prasanth.
Prasanth is a DevOps Engineer.
Prasanth is from Vizag.
```

-
  ```
  ubuntu@ip-172-31-32-223:~$ grep -n Prasanth file3
  ```
  n represents number of lines

```
1:Hi this is Prasanth
2:Prasanth is a DevOps Engineer.
3:Prasanth is from Vizag.
```

egrep(extended grep): Extended version of grep with support for more complex regular expressions.

```
ubuntu@ip-172-31-32-223:~$ egrep "is|Prasanth" file3
Hi this is Prasanth.
Prasanth is a DevOps Engineer.
Prasanth is from Vizag.
```

fgrep(fixed-string grep):

It searches for fixed strings instead of patterns, which means it treats the search string as a literal string rather than interpreting it as a regular expression.

It's useful when you want to search for a specific string without interpreting special characters as part of a regular expression.

fgrep is generally faster than grep or egrep when searching for fixed strings because it doesn't need to interpret regular expressions.

Example: fgrep "string" file.txt searches for the exact string "string" in the file.

```
ubuntu@ip-172-31-32-223:~$ fgrep "Prasanth" file3
Hi this is Prasanth.
Prasanth is a DevOps Engineer.
Prasanth is from Vizag.
```

awk (aho, weinberger, kernighan): A powerful pattern scanning and text processing language that can also be used for searching.

```
# Note: $1 --Employee, $2 --Role, $3 --Salary
ubuntu@ip-172-31-32-223:~$ awk '{print}' file4
Employee   Role   Salary
```

```
Sasikala   Developer   20,800
Prasanth   SystemEngineer   20,800
Satya         Frontend   15,000
Ramu       SystemAdmin   15,000
Varun      Developer  30,000
MuniKumar  UI/UX  18,000
Rupam      Operations    20,000
Vincent    Operations   15,000
```

* I have to find number of Developer in the above table*

```
ubuntu@ip-172-31-32-223:~$ awk '/Developer/{print}' file4
```

o/p

```
Sasikala   Developer   20,800
Varun      Developer   30,000
```

*I must find only employee name and salary from above table *

```
ubuntu@ip-172-31-32-223:~$ awk '{print $1,$3}' file4
Employee   Salary
Sasikala    20,800
Prasanth    20,800
Satya          15,000
Ramu           15,000
Varun           30,000
MuniKumar  18,000
Rupam            20,000
Vincent         15,000
```

*I need to find total number of employees in a company *

```
ubuntu@ip-172-31-32-223:~$ awk '{print NR,$0}' file4
1 Employee   Role   Salary
2 Sasikala   Developer  20,800
3 Prasanth   SystemEngineer   20,800
4 Satya         Frontend    15,000
5 Ramu       SystemAdmin   15,000
6 Varun      Developer  30,000
7 MuniKumar  UI/UX  18,000
8 Rupam      Operations  20,000
9 Vincent    Operations  15,000
```

NOTE **NR** represents new record (Simply identify as total row)

Print the last column of the above given table.

```
ubuntu@ip-172-31-32-223:~$ awk '{print $NF}' file4
Salary
20,800
20,800
15,000
15,000
30,000
18,000
20,000
15,000
```

NOTE **NF** represents new field (Simply identify as last column)

*For example, there are thousands of employees in a company how you will check entire data has been entered or not.

In the above case there are 3 columns Employee, Role & Salary

Now must check whether any column missing.

```
ubuntu@ip-172-31-32-223:~$ awk '{print NR,"->",NF}' file4

1 -> 3

2 -> 3

3 -> 3

4 -> 3

5 -> 3

6 -> 3

7 -> 3

8 -> 3

9 -> 3
```

This all about awk command.

sed: Stream editor for filtering and transforming text. While it's not primarily for searching, it can be used to delete or substitute text based on patterns.

For example, Suppose we have a file named

```
Hello, World!

This is a sample text file.

It contains some example sentences.
```

1.Search and Replace

*I have to find the that 2nd line (sample) would have to change the dummy

```
ubuntu@ip-172-31-32-223:~$ sed 's/sample/dummy/' file5
```

O/P

```
Hello, World!
This is a dummy text file.
It contains some example sentences.
```

2. In place editing

* To perform the replacement directly in the file (in-place editing), use the -i option:

```
ubuntu@ip-172-31-32-223:~$ sed -i 's/sample/demo/' file5
```

O/P

```
Hello, World!
This is a demo text file.
It contains some example sentences.
```

3: Delete Lines Matching a Pattern

Let's delete lines containing the word "example" from the file:

```
ubuntu@ip-172-31-32-223:~$ sed '/example/d' file5
Hello, World!
This is a demo text file.
It contains some example sentences.
```

O/P:

```
Hello, World!
This is a demo text file.
```

4: Insert Text

Let's insert a new line after the first line:

```
ubuntu@ip-172-31-32-223:~$ sed '1 a\This is a new line.' file5
```

O/P:

```
Hello, World!
This is a new line.
This is a demo text file.
It contains some example sentences.
```

5: Append Text

Let's append text at the end of each line:

```
ubuntu@ip-172-31-32-223:~$ sed 's/$/ --End of Line/' file5
```

O/P:

```
Hello, World! --End of Line
This is a demo text file. --End of Line
It contains some example sentences. --End of Line
```

6: Print Specific Lines

Let's print only the second line of the file:

```
ubuntu@ip-172-31-32-223:~$ sed -n '2p' file5
```

O/P:

```
This is a demo text file.
```

cut: Extracts sections from each line of files. It's primarily used to cut out specific fields or columns from text.

-b: specify byte positions for extracting data from files

-c: selected by character

-d: selected by delimeter

-f: selected by filed

*For example, this is file4

```
Employee   Role   Salary
Sasikala   Developer  20,800
Prasanth   SystemEngineer  20,800
Satyanaryana  Frontend  15,000
Ramu       SystemAdmin   15,000
Varun      Developer  30,000
MuniKumar  UI/UX  18,000
Rupam      Operations  20,000
Vincent    Operations  15,000
```

1.**Extract Specific Byte Positions**:

```
ubuntu@ip-172-31-32-223:~$ cut -b 1,2,3,4,5 file4
```

O/P: This command extracts bytes 1,2,3,4 and 5 from each line of the file.

```
Emplo
Sasik
Prasa
```

```
Satya
Ramu
Varun
MuniK
Rupam
Vince
```

2. Extract Bytes from a Range of Positions:

```
ubuntu@ip-172-31-32-223:~$ cut -b1-10 file4
```

O/P: This command extracts bytes 1 to 10 from each line of the file.

```
Employee
Sasikala
Prasanth
Satyanarya
Ramu
Varun
MuniKumar
Rupam
Vincent
```

3.Combining Byte Ranges:

```
ubuntu@ip-172-31-32-223:~$ cut -b1-3,7-9 file4
```

O/P:

```
Empee
Sasla
```

```
Prath
Satary
Ram
Var
Munmar
Rup
Vint
```

4.Extract Range of Characters

Let's extract the first 5 characters of each line:

```
ubuntu@ip-172-31-32-223:~$ sudo cut -c 1-5 file4
Emplo
Sasik
Prasa
Satya
Ramu
Varun
MuniK
Rupam
Vince
```

5. Extract Specific Columns

Let's extract the first and third columns (fields) from the file:

Let’s assume file 6 is

```
Name, Age, City
Ramu, 31, Anakapalle
Prasanth, 28, Visakhapatnam
```

```
Sasikala, 26, Kakinada
Vijju, 28, Kakinada
ubuntu@ip-172-31-32-223:~$ cut -d',' -f1,3 file6
```

O/P:

```
Name, City
Ramu, Anakapalle
Prasanth, Visakhapatnam
Sasikala, Kakinada
Vijju, Kakinada
```

6.Exclude Specific Columns

```
ubuntu@ip-172-31-32-223:~$ cut -d',' --complement -f1 file6
 Age, City
 31, Anakapalle
 28, Visakhapatnam
 26, Kakinada
 28, Kakinada
```

sort: The sort command is used to sort lines of text files alphabetically or numerically.

1.Sort Lines Alphabetically

Let's sort the lines of the file alphabetically:

```
ubuntu@ip-172-31-32-223:~$ sort file4
```

O/P:

```
Employee    Role   Salary
MuniKumar  UI/UX  18,000
Prasanth   SystemEngineer  20,800
```

```
Ramu       SystemAdmin   15,000
Rupam      Operations  20,000
Sasikala    Developer  20,800
Satyanaryana  Frontend  15,000
Varun      Developer  30,000
Vincent    Operations  15,000
```

2.Sort Lines in Reverse Order

Let's sort the lines of the file in reverse alphabetical order:

```
ubuntu@ip-172-31-32-223:~$ sort -r file4
```

O/P:

```
Vincent    Operations  15,000
Varun      Developer  30,000
Satyanaryana  Frontend  15,000
Sasikala    Developer  20,800
Rupam      Operations  20,000
Ramu       SystemAdmin   15,000
Prasanth   SystemEngineer  20,800
MuniKumar  UI/UX  18,000
Employee    Role   Salary
```

*The -r option is used to sort the lines in reverse order (descending).

3. Sort Numerically

Suppose we have a file named numbers.txt with the following content:

```
10
```

```
5
20
15
```

Let's sort the lines numerically:

```
sort -n numbers.txt
```

O/P:

```
5
10
15
20
```

4. Sort and Remove Duplicate Lines

Let's sort the lines of the file and remove duplicate lines:

```
ubuntu@ip-172-31-32-223:~$ sort -u file4
Employee    Role   Salary
MuniKumar  UI/UX  18,000
Prasanth   SystemEngineer 20,800
Ramu       SystemAdmin   15,000
Rupam      Operations  20,000
Sasikala    Developer  20,800
Satyanaryana  Frontend  15,000
Varun      Developer  30,000
Vincent    Operations  15,000
```

a) Sort and Remove Duplicates, Save to Temporary File:

```
 ubuntu@ip-172-31-32-223:~$ sort -u file4  >newfile5
```

b) Overwrite Original File with Temporary File:

```
ubuntu@ip-172-31-32-223:~$ mv newfile5 file4
```

uniq:

For example, taken file 7 as given data below

```
apple
apple
banana
grape
orange
```

a) Count Occurrences of Each Line

To count how many times each line occurs in the sorted file:

```
ubuntu@ip-172-31-32-223:~$ uniq -c file7
      2 apple
      1 banana
      1 grape
      1 orange
```

The -c option prefixes each line with the number of occurrences.

b) Show Only Duplicate Lines

To show only lines that are repeated (and how many times they are repeated):

```
ubuntu@ip-172-31-32-223:~$ uniq -d file7
apple
```

c) Show Only Unique Lines (Non-Repeated Lines)

To show only lines that are not repeated:

```
ubuntu@ip-172-31-32-223:~$ uniq -u file7
banana
grape
orange
```

The -u option shows only unique lines (lines that do not have any duplicates).

tr: Translates or deletes characters. It's used to replace or remove characters from input text.

Example: tr 'a-z' 'A-Z' < file.txt (Convert lowercase to uppercase)

a) Translating Characters

Suppose you have a file named file8 with the following content:

```
ubuntu@ip-172-31-32-223:~$ cat file8
HI I'M PRASANTH REDDY JEERU
```

[To translate all uppercase letters to lowercase]:

```
ubuntu@ip-172-31-32-223:~$ cat file8 | tr 'A-Z' 'a-z'
hi i'm prasanth reddy jeeru
```

How to save the file permanently

```
ubuntu@ip-172-31-32-223:~$ cat file8 | tr 'A-Z' 'a-z' > file9
```

New file update one will be created.

```
mv file9 file8
```

b) Deleting Characters

To delete all vowels from the content:

```
cat file8 | tr -d 'aeiouAEIOU'
```

O/P:

```
ubuntu@ip-172-31-32-223:~$ cat file8 | tr -d 'AEIOU'
H 'M PRSNTH RDDY JR
```

c) Using Character Classes

Suppose you have a file named file8 with the following content:

```
 HI I'M PRASANTH REDDY JEERU
123
ubuntu@ip-172-31-32-223:~$ cat file8 | tr -d '[:digit:]'
```

O/P:

```
 HI I'M PRASANTH REDDY JEERU
```

```
ubuntu@ip-172-31-32-223:~$ cat file8 | tr -d '[:alpha:]'
```

O/P:

```
 '
123
```

EDITORS

vi and nano editors to create, edit, and save files in Linux.

vi

The vi editor is a powerful text editor available on almost all Unix-like operating systems. Here's a complete example of using vi to create and edit a file.

```
ubuntu@ip-172-31-32-223:~$ cat file3
Hi this is Vinay.
```

```
Vinay is a DevOps Engineer.
Vinay is badminton player.
Hi Vinay is from Vizag
ubuntu@ip-172-31-32-223:~$ vi file3
```

Enter Insert Mode and Add Text:

Press i to enter Insert mode, allowing you to type and edit text.

Type Your Content:

Exit Insert Mode:

Press Esc to return to Command mode.

Save and Exit:

Type :wq and press Enter to save changes and exit vi.

nano

```
ubuntu@ip-172-31-32-223:~$ ls
file1 file2 file3 file4 file5
```

There is total 5 files I have to open one file and I have to edit one file

For example, I had to open file3

```
ubuntu@ip-172-31-32-223:~$ nano file3
```

one file will be open there you can edit and **press ctrl+X and Press Y and Enter** to save the file if any changes made.

(or)

press ctrl+O and then press Enter to save the file & ctrl+X to exit.

diff

The diff command compares files line by line and displays the differences between them.

Basic Usage Compare Two Files:

```
ubuntu@ip-172-31-32-223:~$ diff file2 file5
```

O/P:

```
1,3c1,3
< 1.Hi how are you.
< 2.Where are you working
< 60.Working in CtrlAltFix Solutions
---
> Hello, World!
> This is a demo text file.
> It contains some example sentences.
```

< denotes lines from file2

> denotes lines from file5

Show Side-by-Side Comparison:

```
ubuntu@ip-172-31-32-223:~$ diff -y file2 file5
1.Hi how are you.                                    | Hello, World!
2.Where are you working                            | This is a demo
text file.
60.Working in CtrlAltFix Solutions           | It contains some
                                                              examples
sentences.
```

cmp:

a) Basic Usage Compare Two Files:

```
ubuntu@ip-172-31-32-223:~$ cmp file3 file5
```

O/P:

```
file3 file5 differ: byte 2, line 1
```

b) Show All Differences

```
ubuntu@ip-172-31-32-223:~$ cmp -l file3 file5
```

This output lists all differing bytes and their positions in the two files.

File Permissions and Ownership

chmod, chown, and chgrp commands, which are used to change file permissions, ownership, and group ownership respectively**.**

Table**:**

r w x

0 0 0 ->0

0 0 1 ->1

0 1 0 ->2

0 1 1 ->3

1 0 0 ->4

1 0 1 ->5

1 1 0 ->6

1 1 1 ->7

read -4

write - 2

Execute -1

chmod - Change File Permissions

File permissions in Linux are represented as a set of read (r), write (w), and execute (x) permissions for the user (owner), group, and others.

Example Setup:

Suppose you have a file named file1 with the following initial permissions:

```
-rw-r--r-- 1 root   root      77 May  6 09:38 file1
```

Command Examples:

a. Change Permissions Using Symbolic Mode:

1)To add execute permissions for the user:

```
sudo chmod u+x file1 {first}
```

Resulting permissions:

```
-rwxr--r-- 1 root   root      77 May  6 09:38 file1
```

2) To remove write permissions for the group

```
sudo chmod g-w file1 {middle}
```

Resulting permissions:

```
-rwxr----- 1 root root  77 May  6 09:38 file1
```

3) To add read permissions for others: {**last** }

```
sudo chmod o+r file1
```

b) Change Permissions Using Octal Mode:

chmod

To set permissions to rwxr-xr-x (755 in octal):

```
ubuntu@ip-172-31-32-223:~$ sudo chmod 755 file1
```

Resulting permissions:

```
-rwxr-xr-x 1 root   root      77 May  6 09:38 file1
```

To set permissions to rw-rw-r-- (664 in octal):

```
ubuntu@ip-172-31-32-223:~$ sudo chmod 664 file1
```

Resulting permissions:

```
-rw-rw-r-- 1 root   root      77 May  6 09:38 file1
```

chown - Change File Ownership

chown is used to change the owner and optionally the group of a file.

Example Setup:

Suppose you have a file named file1 owned by user user1 and group group1.

Command Examples:

a. Change Owner:

To change the owner to prasanth:

```
chown prasanth file1
```

(or) `sudo chown -c prasanth file1`

b. Change Owner and Group:

To change the owner to prasanth and the group to root:

```
chown prasanth:root file1
```

c. Change Group only

To change the group to root:

```
chown  :root file1
```

chgrp - Change Group Ownership

chgrp is used to change the group ownership of a file.

Example Setup:

Suppose you have a file named file1 owned by group group1.

Command Examples:

a. Change Group:

To change the group to root

```
chgrp root file1
```

(or)

```
sudo chgrp -c root file1
```

Access Control Lists (ACLs)

ACLs provide a more flexible permission mechanism than the basic file permissions. They allow setting permissions for individual users and groups beyond the owner, group, and others.

Use the `getfacl` command to view ACLs of a file:

```
ubuntu@ip-172-31-32-223:~$ getfacl file4

# file: file4

# owner: ubuntu

# group: ubuntu

user::rw-

group::rw-

mask::rw-

other::r—
```

Adding a User ACL

```
ubuntu@ip-172-31-32-223:~$ sudo setfacl -m u:hanuman:rw file4
ubuntu@ip-172-31-32-223:~$ getfacl file4
# file: file4
# owner: ubuntu
# group: ubuntu
user::rw-
user:hanuman:rw-
group::rw-
mask::rw-
other::r—
```

Adding a Group ACL

```
ubuntu@ip-172-31-32-223:~$ setfacl -m g:hanuman:rx file4
ubuntu@ip-172-31-32-223:~$ getfacl file4
# file: file4
# owner: ubuntu
# group: ubuntu
user::rw-
group::rw-
group:hanuman:r-x
mask::rwx
other::r—
```

Removing an ACL

```
ubuntu@ip-172-31-32-223:~$ setfacl -x g:hanuman file4
ubuntu@ip-172-31-32-223:~$ getfacl file4
# file: file4
```

# owner: ubuntu

# group: ubuntu

user::rw-

group::rw-

mask::rw-

other::r—

User Management

Create a New User**:**

useradd

sudo useradd hanuman

This creates a new user named `username` with default settings.

To display where the user has created or not.

(Or)

sudo useradd -m -d /home/hanuman -s /bin/bash hanuman

→ To add the user for existing user and also it creates a new user

ubuntu@ip-172-31-32-223:~$ sudo useradd -G prasanth kalyan

To see the user and group:

ubuntu@ip-172-31-32-223:~$ cat /etc/passwd

O/P

prasanth:x:1001:1001::/home/prasanth:/bin/sh

hanuman:x:1002:1002::/home/hanuman:/bin/sh

hanumanji:x:1003:1003::/home/hanumanji:/bin/bash

```
ubuntu@ip-172-31-32-223:~$ getent passwd
```

O/P

```
prasanth:x:1001:1001::/home/prasanth:/bin/sh
hanuman:x:1002:1002::/home/hanuman:/bin/sh
hanumanji:x:1003:1003::/home/hanumanji:/bin/bash
```

```
ubuntu@ip-172-31-32-223:~$ cat /etc/group
```

O/P

```
prasanth:x:1001:
hanuman:x:1002:
hanumanji:x:1003:
```

How to identify if a user is created?

id

```
ubuntu@ip-172-31-32-223:~$ id hanuman
uid=1002(hanuman) gid=1002(hanuman) groups=1002(hanuman)
```

Note: uid → userid

gid → groupid

usermod

```
ubuntu@ip-172-31-32-223:~$ sudo usermod -l prasanthreddy prasanth
```

sudo usermod -l newname oldname

Add the User to the sudo Group:

```
sudo usermod -aG sudo prasanth
```

Verify the Change:

```
su - prasanth
```

Lock the User Account

```
sudo usermod -L Prasanth
```

Unlock the user account

```
sudo usermod -U Prasanth
```

passwd

The passwd command is used to change the password of a user account.

Change Your Own Password:

```
ubuntu@ip-172-31-32-223: ~$ passwd
Changing password for ubuntu.
Current password:
```

You will be changed to enter your current password and then the new password.

Change Another User's Password:

```
ubuntu@ip-172-31-32-223: ~$ sudo passwd prasanth
New password:
Retype new password:
passwd: password updated successfully
```

Using passwd -S

The `passwd -S` command provides a status summary of a user's account.

```
ubuntu@ip-172-31-32-223:~$ sudo passwd -S prasanth
prasanth L 06/10/2024 0 99999 7 -1
```

If the account is locked, it will show L for locked.

```
ubuntu@ip-172-31-32-223:~$ sudo passwd -S prasanth
prasanth P 06/10/2024 0 99999 7 -1
```

If the account is unlocked, it will indicate P.

Checking the Shadow File

You can also check the /etc/shadow file directly. If a user's account is locked, there will be a ! or * at the beginning of the encrypted password field.

```
ubuntu@ip-172-31-32-223: ~$ sudo grep prasanth /etc/shadow
prasanth:$y$j9T$ZaPX68fPi3zmbW2ye6Myw1$aWnTabLBzM8NUKo1mldEKpsCb5dcubdMM5ELqllLG8.:19884:0:99999:7:::
```

→The ! before the hashed password indicates that the account is locked.

→ If there is no ! or *, the account is not locked.

chage

The chage command can be used to:

-l: Display the password aging information.

-m: Set the minimum number of days between password changes.

-M: Set the maximum number of days the password is valid.

-W: Set the number of days of warning before password expiration.

-I: Set the number of inactive days after password expiration.

-E: Set the account expiration date.

```
ubuntu@ip-172-31-32-223:~$ sudo chage -l hanuman
Last password change                              : May 30, 2024
Password expires                                    : never
Password inactive                                  : never
Account expires                                     : never
Minimum number of days between password change          : 0
Maximum number of days between password change          :
99999
Number of days of warning before password expires       : 7
```

```
ubuntu@ip-172-31-32-223:~$ sudo chage -m 7 -M 90 -W 10 -I
30 -E 2025-01-31 prasanth
ubuntu@ip-172-31-32-223:~$ sudo chage -l prasanth
Last password change                              : Jun 10, 2024
Password expires                                    : Sep 08, 2024
Password inactive                                  : Oct 08, 2024
Account expires                                     : Jan 31, 2025
Minimum number of days between password change     : 7
Maximum number of days between password change     : 90
Number of days of warning before password expires       : 10
```

userdel

a) To delete a user Reddy without removing their home directory:

```
ubuntu@ip-172-31-32-223:~$ sudo userdel Reddy
```

b) Verify the user is deleted by searching for the username in the /etc/passwd

```
ubuntu@ip-172-31-32-223: ~$ grep Reddy /etc/passwd
```

This command should return nothing if the user `Reddy` has been successfully deleted.

c)To delete the user Reddy and remove their home directory:

```
   ubuntu@ip-172-31-32-223:~$ sudo userdel -r Reddy
```

Adding and Deleting Users

adduser

is used for creating new user accounts with prompts for additional information.

a) Add User john:

```
sudo adduser janu
```

b) Add User john to Group sudo:

```
sudo adduser janu prasanth
```

c) check the groups:

```
ubuntu@ip-172-31-32-223: ~$ groups janu
janu : janu prasanth
```

deluser

is used for deleting user accounts and optionally their home directories and associated files.

a) To delete the user john and their home directory:

```
ubuntu@ip-172-31-32-223: ~$ sudo deluser --remove-home janu
```

b) Verify User Deletion:

```
ubuntu@ip-172-31-32-223: ~$ grep john /etc/passwd
```

This command should return nothing if the user janu has been successfully deleted.

Group Management

groupadd

a) To create a new group

```
ubuntu@ip-172-31-32-223: ~$ sudo groupadd testgroup
```

b) Create a New Group with a Specific GID:

To create a new group called designers with a GID of 2000:

```
ubuntu@ip-172-31-32-223: ~$ sudo groupadd -g 2000 designers
```

groupmod

Used to modify existing groups (change GID or rename the group)

a) Modify Group GID:

To change the GID of testgroup to 3000:

```
ubuntu@ip-172-31-32-223:~$ sudo groupmod -g 3000 testgroup
```

b) Rename the Group:

```
sudo groupmod -n prasanthgroup testgroup
```

groupdel

Used to delete groups from the system.

```
ubuntu@ip-172-31-32-223:~$ sudo groupdel designers
```

Process Management

ps

The ps command is used to display information about active processes.

```
ubuntu@ip-172-31-32-223:~$ ps
    PID TTY          TIME CMD
   1339 pts/0    00:00:00 bash
  1577      /0    00:00:00 ps
```

Viewing Processes:

a) Detailed list of all processes:

```
ubuntu@ip-172-31-32-223:~$ ps aux
```

a: Show processes for all users.

u: Display the user-oriented format.

x: Show processes not attached to a terminal.

b) Display processes for the current user in a simple format:

```
ubuntu@ip-172-31-32-223:~$ ps -u
USER      PID     %CPU   %MEM    VSZ   RSS     TTY      STAT START
TIME CMD
ubuntu   2719   0.0       0.5           9152  5248  pts/0    Ss   08:25
0:00  -bash
ubuntu   2966   0.0       0.3         10464  3200  pts/0    R+   09:07    0:00
ps -u
```

c) Tree view of processes:

```
ubuntu@ip-172-31-32-223:~$ ps -ejH
```

e: Show all processes.

j: Display the job control format.

H: Show the process hierarchy (tree format).

(or)

```
ubuntu@ip-172-31-32-223:~$ ps -e –forest
```

d) Display processes by a specific user:

```
ubuntu@ip-172-31-32-223:~$ ps -u username
```

Replace username with the actual username.

e) Display processes by process ID (PID):

```
ubuntu@ip-172-31-32-223:~$ ps -p 591
PID TTY          TIME CMD
591 ?        00:00:00 nginx
```

f) Display processes with a specific name:

```
ubuntu@ip-172-31-32-223:~$ ps -C nginx
    PID TTY          TIME CMD
    591 ?        00:00:00 nginx
    594 ?        00:00:00 nginx
```

g) Sort processes by memory usage:

```
ubuntu@ip-172-31-32-223:~$ ps aux --sort=-%mem
```

top

a) Simply type top and press Enter.

```
ubuntu@ip-172-31-32-223:~$ top
```

This will display a dynamic, real-time view of the system processes.

b) Run `top` with Specific User:

```
ubuntu@ip-172-31-32-223:~$ top -u root
```

htop

htop: An interactive process viewer (requires installation)

If the process is non-essential, press F9 and choose SIGTERM to kill it gracefully.

If the process is important but needs to be deprioritized, press F8 to increase its nice value, lowering its priority.

{https://www.youtube.com/watch?v=QBsH5bkkXyg&ab_channel=LinuxTutorial }

pgrep

pgrep: Searches for processes based on name and other attributes.

```
ubuntu@ip-172-31-32-223:~$ pgrep nginx
591
594
```

Controlling Processes:

kill

It is used to terminate a process manually.

a) To see all the signal names

```
ubuntu@ip-172-31-32-223:~$ kill -l
```

b) Replace `PID` with the process ID you want to terminate.

```
ubuntu@ip-172-31-32-223:~$ kill PID
```

kill -1 PID (To restart the process)

kill -2 PID (Interrupt from keyboard like Ctrl+c)

kill -9 PID (Forcefully terminate the process)

kill -15 PID (kill process gracefully)

c)killall: Sends a signal to all processes matching a name.

```
ubuntu@ip-172-31-32-223:~$ killall nginx
```

d)pkill: Sends a signal to processes based on name and other attributes.

```
ubuntu@ip-172-31-32-223:~$ pkill -9 nginx
```

The -9 option sends the SIGKILL signal, forcefully terminating the process.

Monitoring and Managing Processes

Managing Process:

nice Niceness scale goes from -20 to 19. The lower the number more priority that tasks gets.

Process priority = nice

(ex: nice -n 5 process)

```
ubuntu@ip-172-31-32-223:~$ ps -ef | grep ./Notify_SMS.sh

root       2433       1  0 Jun20 ?        00:00:00 sudo nice -n -2
./Notify_SMS.sh

root       2434    2433  0 Jun20 ?        00:00:00 sudo nice -n -2
./Notify_SMS.sh

root       2435    2434  0 Jun20 ?        00:00:00 /bin/bash
./Notify_SMS.sh

ubuntu     10908   10907  0 06:36 ?        00:00:00 /bin/bash
/home/ubuntu/Notify_SMS.sh

ubuntu     10911   10293  0 06:36 pts/2    00:00:00 /bin/bash
./Notify_SMS.sh
```

```
ubuntu     10933   10932  0 06:40 ?        00:00:00 /bin/sh -c
/home/ubuntu/Notify_SMS.sh
ubuntu     10938   10293  0 06:40 pts/2    00:00:00 grep --color=auto
./Notify_SMS.sh
```

```
ubuntu@ip-172-31-32-223:~$ sudo kill -9 2435
```

I have killed the root user .

```
ubuntu@ip-172-31-32-223:~$ ps -l 11489
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME
CMD
0 S  1000   11489   11488  0  80   0 -  1941 do_wai ?          0:00
/bin/bash /home/ubuntu/Notify_SMS.sh
```

nice: Starts a process with a specified priority.

*Nice Values Range: The nice value ranges from -20 (highest priority) to 19 (lowest priority).

*Default Nice Value: By default, processes start with a nice value of 0.

The nice command is used to start a new process with a specified nice value. Here's the syntax:

```
nice -n [nice_value] [command]
```

1.Starting a Process with a Higher Priority:

To start a process with a lower nice value (higher priority), you usually need root privileges. For example, to start a process with a nice value of -5:

```
ubuntu@ip-172-31-32-223:~$ sudo nice -n -5 ./Notify_SMS.sh
```

2. Starting a Process with a Lower Priority:

To start a process with a higher nice value (lower priority), for example, 10:

```
ubuntu@ip-172-31-32-223:~$ sudo nice -n 10 ./Notify_SMS.sh
```

renice: Changes the priority of an already running process.

```
ubuntu@ip-172-31-32-223:~$ ps -l 13449
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME
CMD
0 S  1000   13449   13448  0  80   0 -  1941 do_wai ?          0:00
/bin/bash /home/ubuntu/Notify_SMS.sh
```

{For example, here we will see the PID is 13449 and NI (nice value is 0)}

```
ubuntu@ip-172-31-32-223:~$ sudo renice -n -8 -p 13449
13449 (process ID) old priority 0, new priority -8
ubuntu@ip-172-31-32-223:~$ ps -l 13449
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME
CMD
0 S  1000   13449   13448  0  72  -8 -  1941 do_wai ?          0:00
/bin/bash /home/ubuntu/Notify_SMS.sh
```

Job Control:

jobs

jobs: Lists the jobs running in the background in the current shell session.

```
ubuntu@ip-172-31-32-223:~$ jobs
[1]   Stopped                 ./Notify_SMS.sh
[2]   Stopped                 sleep 20s
[3]-  Stopped                 sleep 50s
[4]+  Stopped                 nice -n 9 ./Notify_SMS.sh
```

For example, I have four jobs running in the background.

(Now I had to run the second row. Let’s see how)

bg Resumes a suspended job in the background.

```
ubuntu@ip-172-31-32-223:~$ bg %2
[2] sleep 20s &
```

```
ubuntu@ip-172-31-32-223:~$ jobs
[1]   Stopped                 ./Notify_SMS.sh
[2]   Done                    sleep 20s
[3]-  Stopped                 sleep 50s
[4]+  Stopped                 nice -n 9 ./Notify_SMS.sh
```

In the above while we given jobs command all 4 rows in stopped condition. But we will observe once that the 2 row is done.

That means it runs in background by giving **bg %2** command.

fg Resume job to the foreground.

```
ubuntu@ip-172-31-32-223:~$ fg %4
nice -n 9 ./Notify_SMS.sh
```

```
ubuntu@ip-172-31-32-223:~$ jobs
[1]-  Stopped                 ./Notify_SMS.sh
[3]+  Stopped                 sleep 50s
```

nohup:

Runs a command immune to hangups, with output to a non-tty.

(example: nohup command &)

The & puts the process in the background.

```
ubuntu@ip-172-31-32-223:~$ jobs
```

First, I checked whether any jobs are running or not.

Now I had to open two command prompts at a time, I will schedule a one job on same instance. After that I will again run the jobs in both cmd’s

And close one cmd then the job will disappear on other side cmd if we check jobs.

For that cause we must use the command given below:

```
*ubuntu@ip-172-31-32-223:~$ nohup ./echo.sh &
```

To check job

```
ps -ef | grep echo.sh
```

Now if you close one cmd then also it will run in other cmd.

Process Information:

pstree: Displays a tree of processes.

```
ubuntu@ip-172-31-32-223:~$ pstree
```

b) If we run the command pstree $$ we are going to see a very simple display

```
ubuntu@ip-172-31-32-223:~$ pstree $$
bash———pstree
```

c)All the process being run by the systemd: The pstree command allow you to view the process.

```
ubuntu@ip-172-31-32-223:~$ pstree root |head -5
systemd-+-acpid
        |-2*[agetty]
        |-amazon-ssm-agen---7*[{amazon-ssm-agen}]
        |-chronyd---chronyd
        |-cron---2*[cron---sh---Notify_SMS.sh-+-curl]
```

d)with the username PID’s

```
ubuntu@ip-172-31-32-223:~$ pstree -p ubuntu
sh(11509)───Notify_SMS.sh(11510)─┬─curl(11981)
                                 └─grep(11982)
sh(11976)───Notify_SMS.sh(11977)─┬─curl(11978)
                                 └─grep(11979)
sshd(11945)───bash(11946)───pstree(11990)
systemd(11863)───(sd-pam)(11864)
```

e) with PID'S it will show the process

```
ubuntu@ip-172-31-32-223:~$ pstree -s 12119
systemd───cron───cron───sh───Notify_SMS.sh─┬─curl
                                           └─grep
```

f) with PID's it will show all

```
ubuntu@ip-172-31-32-223:~$ pstree -ps 12119
systemd(1)───cron(354)───cron(12117)───sh(12118)───Notify_SMS.sh(12119)─┬─curl(12132)
                                                                        └─grep(12133)
```

lsof: Lists open files associated with processes.

a) To properly list all open files, you use sudo. Otherwise into a lot of permission denied warnings.

```
ubuntu@ip-172-31-32-223:~$ sudo lsof
```

b) To see how many files are there.

```
ubuntu@ip-172-31-32-223:~$ sudo lsof |wc -l
2846
```

strace: Traces system calls and signals of a process.

Example: strace -p PID

Replace PID with the process ID you want to trace.

```
ubuntu@ip-172-31-32-223:~$ sudo strace -p 440
strace: Process 440 attached
gettimeofday({tv_sec=1719306326, tv_usec=861568}, NULL) = 0
epoll_wait(9,
```

If you press ctrl+c it will be detached.

```
ubuntu@ip-172-31-32-223: ~$ strace -o file2 ls -l
```

{Intercepting System Calls: strace will intercept all the system calls made by the ls -l command. System calls are the fundamental interface between an application and the Linux kernel. They perform tasks such as reading files, writing to files, opening sockets, creating processes, etc.

Logging to File: Instead of displaying the intercepted system calls in the terminal, strace will write them to file2.

Executing the Command: The ls -l command will be executed as usual, listing the files in the current directory in long format, showing details such as permissions, number of links, owner, group, size, and timestamp.}

```
ubuntu@ip-172-31-32-223:~$ vim file2
```

this detailed log of system calls, search for specific calls, and better understand the behavior of the `ls -l` command as it interacts with the Linux kernel.

```
ubuntu@ip-172-31-32-223:~$ strace -c ls -l
```

The strace -c ls -l command is a powerful tool for summarizing system call statistics for a given command. It provides a concise overview of the system calls made, how often they were called, and how much time was spent in each call. This can be extremely useful for performance tuning and debugging.

watch:

watch: Runs a command repeatedly, displaying its output and errors.

watch free -h.

```
ubuntu@ip-172-31-32-223:~$watch free -h
```

```
Every 2.0s: free -h
                                                    ip-172-31-32-223: Tue Jun 25
10:19:41 2024

              total        used        free      shared       buff/cache   available
Mem:      949Mi       186Mi    135Mi       0.0Ki       627Mi          598Mi
Swap:       0B            0B            0B
```

```
ubuntu@ip-172-31-32-223:~$ watch -n 1 df -h
Every 1.0s: df -h                          ip-172-31-32-223: Tue Jun 25
10:28:17 2024

Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  4.2G  3.4G  56% /
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  868K  190M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
```

General System Information

uname: Prints system information.

```
ubuntu@ip-172-31-32-223:~$ uname -a
Linux ip-172-31-32-223 6.5.0-1020-aws #20~22.04.1-Ubuntu SMP Wed
May  1 16:10:50 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

-a: Displays all system information.

Other options: -r (kernel release), -m (machine hardware name), -n (nodename).

hostnamectl: Control the system hostname and get related information.

```
ubuntu@ip-172-31-32-223:~$ hostnamectl
 Static hostname: ip-172-31-32-223
       Icon name: computer-vm
         Chassis: vm
      Machine ID: e1454e55d1424854ae6a0ed4a0288fbf
         Boot ID: 4e8f145a92314d31ab5db6fc9d9b212b
  Virtualization: xen
Operating System: Ubuntu 22.04.4 LTS
          Kernel: Linux 6.5.0-1020-aws
    Architecture: x86-64
 Hardware Vendor: Xen
  Hardware Model: HVM domU
```

This command provides details about the hostname, operating system, kernel, architecture, etc.

System Monitoring and Performance

Memory Information:

free: Displays the amount of free and used memory in the system.

```
ubuntu@ip-172-31-32-223:~$ free -h
               total        used        free      shared  buff/cache
available
Mem:       949Mi       188Mi      125Mi       0.0Ki       635Mi
597Mi
Swap:         0B          0B            0B
```

Note   -h: Human-readable format.

b) Keep refreshing memory after N sec

```
ubuntu@ip-172-31-32-223:~$ free -h -s 2
```

To stop this, click ctrl+c

c)Exit after repeating N times

```
ubuntu@ip-172-31-32-223:~$ free -h -c 2
```

vmstat: Reports virtual memory statistics. It provides information about system processes, memory, paging, block I/O, traps, and CPU activity.

```
ubuntu@ip-172-31-32-223:~$ vmstat

procs -----------memory---------- ---swap-- -----io---- -system-- -----
-cpu-----

 r  b   swpd   free   buff  cache      si   so  bi   bo   in   cs us sy id
wa  st

 0  0      0 128576  55508 595480 0    0     5     9   31   24  0    0
100  0    0
```

→The command vmstat 1 specifically means that vmstat will display this information every second, continuously, until you stop it (usually with Ctrl+C).

```
ubuntu@ip-172-31-32-223:~$ vmstat 1

procs -----------memory---------- ---swap-- -----io---- -system-- -----
-cpu-----

 r  b  swpd   free   buff   cache      si   so    bi    bo   in    cs us sy
id wa st
```

```
 0  0      0 200976  39044 547376    0    0     8    10   32   26  0  0
100  0 0

 0  0      0 200976  39044 547376    0    0     0     0   50   28  0  0
100   0  0

 0  0      0 200976  39044 547376    0    0     0     0   40   30  0  0
100   0  0

 0  0      0 200976  39044 547376    0    0     0     0   46   31  0  0
100  0  0

 1  0      0 200976  39044 547376    0    0     0     0   38   30  0  0
100  0  0
```

In the above command (vmstat 1) the number specifies the interval (in seconds) at which vmstat will refresh and display the system statistics. In this case, it will update every second.

```
ubuntu@ip-172-31-32-223:~$ vmstat  -a

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----

 r  b   swpd  free     inact        active    si   so    bi    bo   in   cs us sy  id
wa  st

 0  0      0    128576  524772  141256    0    0     5     9   31   24  0   0
100  0   0
```

inactive: The process which will executive and the new process which will not be executive this memory is called inactive.

active: The process which are currently running & which are consuming some memory is called active.

```
-a, --active           active/inactive memory

 -f, --forks           number of forks since boot

 -m, --slabs            slabinfo

 -n, --one-header       do not redisplay header

 -s, --stats           event counter statistics

 -d, --disk            disk statistics
 -D, --disk-sum         summarize disk statistics
 -p, --partition <dev> partition specific statistics
 -S, --unit <char>      define display unit
 -w, --wide             wide output
 -t, --timestamp        show timestamp
```

System Resource Usage:

iostat

Reports CPU statistics and I/O statistics for devices and partitions.

To install → sudo apt install sysstat

a)

```
ubuntu@ip-172-31-32-223:~$ iostat
Linux 6.5.0-1022-aws (ip-172-31-32-223)        07/04/24        _x86_64_        (1 CPU)


avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.04    0.01    0.03    0.02    0.01   99.89


Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn
kB_dscd
loop0            0.00         0.00         0.00         0.00         15          0
0
loop1            0.00         0.00         0.00         0.00         32          0
0
loop2            0.00         0.00         0.00         0.00         18          0
0
loop3            0.00         0.02         0.00         0.00       2048          0
0
loop4            0.00         0.00         0.00         0.00         16          0
0
loop5            0.00         0.00         0.00         0.00         57          0
0
loop6            0.00         0.00         0.00         0.00         16          0
0
```

```
loop7          0.00       0.02       0.00                 0.00             1340       0
0

loop8          0.00       0.00       0.00                 0.00               14          0
0

xvda           1.21       7.64       9.59                 0.00               657679
824833      0
```

b) The -x option provides extended statistics, and the 1 specifies the interval in seconds for updating the statistics.

```
ubuntu@ip-172-31-32-223:~$ iostat -x 1
```

Every second, a new report will be generated, showing real-time statistics for the past second.

If you see high values in %util, it indicates that the device is heavily utilized.

High r_await or w_await values indicate latency in read or write operations, suggesting a possible bottleneck.

The r/s and w/s columns help you understand the load on each device in terms of read and write requests.

sar

The **sar** (System Activity Reporter) command in Linux is a powerful tool for collecting, reporting, and saving system activity information. It can report on various system performance metrics such as CPU utilization, memory usage, disk I/O, network activity, and more.

To install → sudo apt-get install sysstat

a) To monitor disk, I/O statistics:

```
ubuntu@ip-172-31-32-223:~$ sar -d 1 10
```

b) To monitor network activity:

```
ubuntu@ip-172-31-32-223:~$ sar -n DEV 1 5

Linux 6.5.0-1022-aws (ip-172-31-32-223)       07/04/24       _x86_64_       (1 CPU)

09:27:03       IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s
rxmcst/s   %ifutil

09:27:04         lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

```
09:27:04        eth0      2.00      0.00      0.10      0.00      0.00      0.00      0.00      0.00
…………..
```

c)To monitor Memory Usage

```
ubuntu@ip-172-31-32-223:~$ sar -r 1 4
Linux 6.5.0-1022-aws (ip-172-31-32-223)        07/04/24       _x86_64_
(1 CPU)

09:32:24    kbmemfree   kbavail kbmemused  %memused kbbuffers
kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
09:32:25      191820    615236    142100    14.62     42108    502508
803216     82.63    288540    313008       44
09:32:26      191820    615236    142100    14.62     42108    502508
803216     82.63    288540    313008       44
Average:      191820    615236    142100    14.62     42109    502507
803216     82.63    288540    313008       44
ubuntu@ip-172-31-32-223:~$
```

d) To monitor CPU usage in real-time at 1-second intervals for a total of 10 intervals:

```
ubuntu@ip-172-31-32-223:~$ sar -u 1 10
Linux 6.5.0-1022-aws (ip-172-31-32-223)        07/04/24       _x86_64_       (1 CPU)

08:57:13        CPU     %user %nice   %system   %iowait   %steal    %idle
08:57:14        all     0.00     0.00     0.00            0.00     0.00          100.00
08:57:15        all     0.99     0.00     0.00            0.00     0.00           99.01
08:57:16        all     0.00     0.00     0.00            0.00     0.00         100.00
08:57:17        all     0.00     0.00     0.99            0.00     0.00           99.01
08:57:18        all     0.00     0.00     0.00            0.00     0.00          100.00
08:57:19        all     0.00     0.00     0.00            0.00     0.00          100.00
08:57:20        all     0.00     0.00     0.00            0.00     0.00          100.00
```

```
08:57:21      all      0.00      0.00      0.00               0.00      0.00         100.00
08:57:22      all      0.00      0.00      0.00               0.00      0.00         100.00
08:57:23      all      0.00      0.00      0.00                0.00      0.00        100.00
Average:       all      0.10      0.00      0.10                 0.00      0.00         99.80
```

***** Example: Enabling sysstat for Continuous Monitoring:

```
sudo nano /etc/default/sysstat
```

Set ENABLED **to** true:

```
sudo systemctl restart sysstat
```

By enabling continuous monitoring, sar will collect and log system activity data at regular intervals, which you can later analyze using various sar.

uptime

The uptime command in Linux is used to tell how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

```
ubuntu@ip-172-31-32-223:~$ uptime
 11:57:00 up 1 day,  5:05,  1 user,  load average: 0.00, 0.00, 0.00
```

Disk and Filesystem Management

Disk Usage:

df: Reports file system disk space usage.

```
ubuntu@ip-172-31-32-223:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  4.2G  3.4G  56% /
tmpfs           475M     0  475M   0% /dev/shm
tmpfs           190M  872K  190M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
```

```
ubuntu@ip-172-31-32-223:~$ df -h -BK
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/root       7941576K 4390880K  3534312K  56% /
tmpfs            486024K       0K   486024K   0% /dev/shm
tmpfs            194412K     872K   193540K   1% /run
tmpfs              5120K       0K     5120K   0% /run/lock
/dev/xvda15      106832K    6186K   100646K   6% /boot/efi
tmpfs             97204K       4K    97200K   1% /run/user/1000
```

*Note: -h stands for human readable format.

K stands for kilobytes

G stands for gigabytes

du: Estimates file space usage.

```
ubuntu@ip-172-31-32-223:~$ du -h
12K     ./.aws
4.0K    ./.config/procps
8.0K    ./.config/htop
16K     ./.config
4.0K    ./.cache/JNA/temp
8.0K    ./.cache/JNA
12K     ./.cache
16K     ./.ssh
4.0K    ./.local/share/nano
8.0K    ./.local/share
12K     ./.local
4.0K    ./project_sasi
212K    .
```

```
ubuntu@ip-172-31-32-223:~$ du -ah /home/ubuntu
4.0K    /home/ubuntu/.aws/config
4.0K    /home/ubuntu/.aws/credentials
12K     /home/ubuntu/.aws
24K     /home/ubuntu/file2
```

*****Note: a stand for all

-h stand for human readable format.

```
ubuntu@ip-172-31-32-223: ~$ du -ahc /home/ubuntu
4.0K    /home/ubuntu/file7
4.0K    /home/ubuntu/.selected_editor
4.0K    /home/ubuntu/echo.sh
16K     /home/ubuntu/.file2.swo
4.0K    /home/ubuntu/Notify_SMS.sh
4.0K    /home/ubuntu/file5
212K     /home/ubuntu/
212K     total
```

*****Note: c stand for count.

```
ubuntu@ip-172-31-32-223:~$ du -h file8
4.0K    file8
ubuntu@ip-172-31-32-223:~$ du -sh file8
4.0K    file8
```

-s: Summary (total size).

-h: Human-readable format.

Disk Partitioning:

fdisk: Partition table manipulator for Linux. And for weave the partitions in Linux servers & Linux desktop computers.

```
ubuntu@ip-172-31-32-223:~$ sudo fdisk -l
```

It will list your partitions here.

lsblk: Lists information about all available or specified block devices.

```
ubuntu@ip-172-31-32-223:~$ lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7983
loop1      7:1    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop3      7:3    0 55.7M  1 loop /snap/core18/2823
loop4      7:4    0 63.9M  1 loop /snap/core20/2264
loop5      7:5    0 63.9M  1 loop /snap/core20/2318
loop6      7:6    0   87M  1 loop /snap/lxd/27428
loop7      7:7    0   87M  1 loop /snap/lxd/28373
loop8      7:8    0 38.7M  1 loop /snap/snapd/21465
loop9      7:9    0 38.8M  1 loop /snap/snapd/21759
loop10     7:10   0 55.7M  1 loop /snap/core18/2829
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0  7.9G  0 part /
├─xvda14 202:14   0    4M  0 part
└─xvda15 202:15   0  106M  0 part /boot/efi
```

Loop Devices (loop0 to loop9): These are pseudo-devices often used for mounting files like snap packages.

Disk Device (xvda): This is your main disk with its partitions (xvda1, xvda14, xvda15).

```
ubuntu@ip-172-31-32-223:~$ lsblk -f
```

we are going to use the flag that you will get some additional information associated with these partitions on Linux.

hwinfo

hwinfo is a command-line utility that collects and prints information about the system hardware. It provides a detailed report for almost all system components in a modular fashion.

a)**hwinfo**

It gather all the hardware information and display here.

```
ubuntu@ip-172-31-32-223:~$ hwinfo
```

b)**hwinfo - -disk**

Instead of displaying all the hardware information

```
ubuntu@ip-172-31-32-223:~$ hwinfo --disk
10: None 00.0: 10600 Disk
  [Created at block.245]
  Unique ID: +b7l.Fxp0d3BezAE
  Parent ID: JaHa.j9MZgu1IFZ6
  SysFS ID: /class/block/xvda
  SysFS BusID: vbd-768
  SysFS Device Link: /devices/vbd-768
  Hardware Class: disk
  Model: "Disk"
  Driver: "vbd"
  Device File: /dev/xvda
  Device Number: block 202:0-202:15
  BIOS id: 0x80
  Drive status: no medium
  Config Status: cfg=new, avail=yes, need=no, active=unknown
  Attached to: #8 (Storage controller)
```

c)hwinfo --short --disk

```
ubuntu@ip-172-31-32-223:~$ sudo hwinfo --short --disk
disk:
  /dev/xvda            Disk

ubuntu@ip-172-31-32-223:~$ sudo hwinfo --short --block
disk:
  /dev/xvda            Disk
partition:
  /dev/xvda1           Partition
  /dev/xvda14          Partition
  /dev/xvda15          Partition
```

unmount:

If you want to unmount one of the snap packages, say loop0 mounted at /snap/amazon-ssm-agent/7983:

```
ubuntu@ip-172-31-32-223:~$ sudo umount /snap/amazon-ssm-agent/7983
```

b) Verify It Is Unmounted

```
ubuntu@ip-172-31-32-223:~$ mount | grep /snap/amazon-ssm-agent
```

If the unmount is successful, there should be no output.

Summary

Unmounting detaches a filesystem from the directory tree.

Mounting attaches a filesystem to a directory tree so that its contents can be accessed.

Loop Devices are typically used for mounting files like ISO images or snap packages.

Disk Devices represent physical disks and their partitions.

mount

```
ubuntu@ip-172-31-32-223:~$ mount | column -t
```

It shows all mounted file systems.

Logical Volume Management (LVM)

LVM is used to manage volume and disk on the Linux server.

Logical volume Manager allows disks to be combined together.

Example of LVM

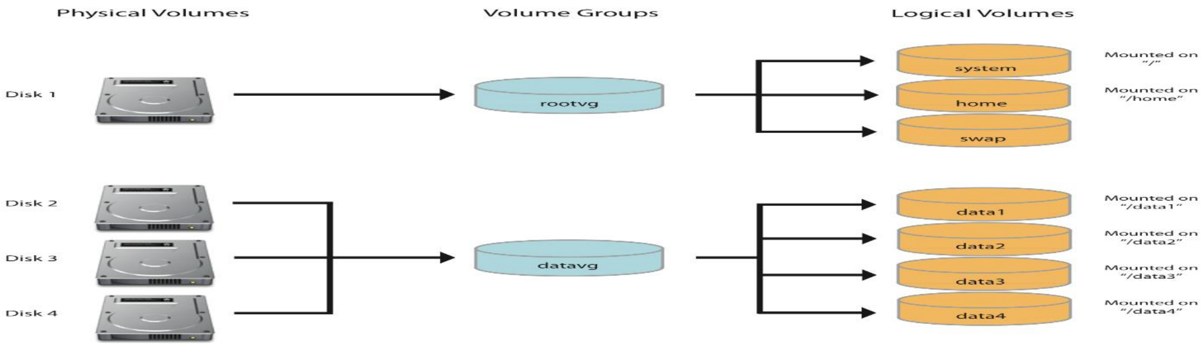
Like partitions of disk in windows C, D drive similarly we can do the same in the linux.

- Single disk can be divided into different partitions.
- Multiple disks combined and group them into the one – then change into different partitions.

Advantage Of LVM

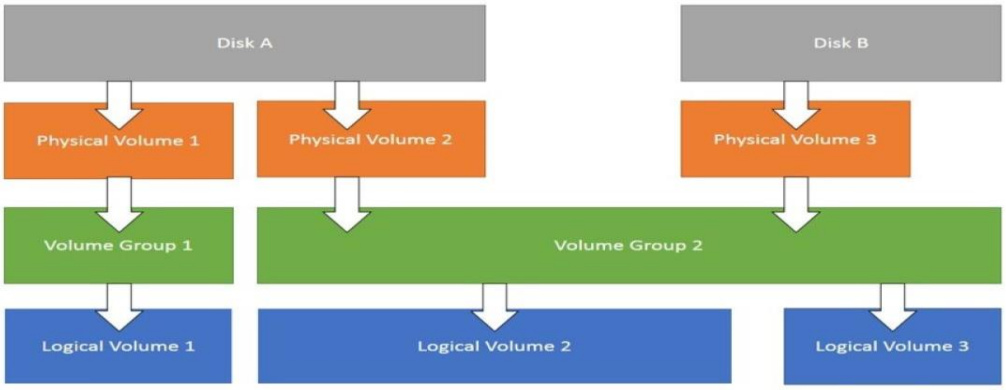
In case of disk is running out of space, you can add a new disk without breaking partitions of file system.

LVM offers flexibility, efficient storage management, improved data safety, easy partitioning, online resizing, performance optimization, and simplified migration.



Possibilities of LVM

- New space can be created on a server for new project.
- In case of low disk increase the space.
- In case of extra space allocated to a partition, capacity can be reallocated (reduce capacity in one volume group and added it to another)



Please refer this video-- https://www.youtube.com/watch?v=FsSf3rmu2Cc

Package Management

Package management is a critical aspect of maintaining and managing software on a Linux system. Different Linux distributions use different package managers. Below are brief descriptions of package management commands for various package managers.

Debian-based Systems (e.g., Ubuntu, Debian)

apt and apt-get

-->Update Package List:

```
sudo apt update
```

-->Upgrade Installed Packages

```
sudo apt upgrade
```

--> Install a Package:

```
sudo apt install package name
```

--> **Remove a Package**

```
sudo apt remove package name
```

--> Search for a Package

```
sudo apt search package name
```

→ Get Package Information

```
sudo apt show package name
```

-->List Installed Packages:

```
apt list –installed
```

dpkg

-->Install a .deb Package:

```
sudo dpkg -i package_file.deb
```

-->Remove a Package:

```
sudo dpkg -r package_name
```

-->List Installed Packages:

```
dpkg -l
```

Red Hat-based Systems (CentOS, Fedora, RHEL)

yum

-->Update Package List:

```
sudo yum check-update
```

-->**Install a Package:**

```
sudo yum install package_name
```

-->Remove a Package:

```
sudo yum remove package_name
```

-->Search for a Package:

```
yum search package_name
```

-->List Installed Packages:

```
yum list installed
```

dnf (for newer systems)

-->Update Package List:

```
sudo dnf check-update
```

-->**Install a Package:**

```
sudo dnf install package_name
```

-->Remove a Package:

```
sudo dnf remove package_name
```

-->Search for a Package:

```
dnf search package_name
```

-->List Installed Packages:

```
dnf list installed
```

rpm

RPM is a software package management system for installing, updating, and removing software packages on Linux-based systems. Red Hat originally developed it but has been adopted by many other Linux distributions. RPM packages, often denoted with the **`.rpm`** file extension contains all the necessary files, metadata, and scripts required to install and manage software on a Linux system.

-->Installing RPM Packages

Suppose you want to install a package named example.rpm

```
rpm -i example.rpm
```

This command will install the `example.rpm` package on your system.

-->Upgrading RPM Packages

```
rpm -U example.rpm
```

This will replace the older version with the newer one.

-->Checking if a Package is Installed using RPM

```
rpm -q example
```

This will display details like the package name, version, and architecture.

-->Verifying RPM Package

```
rpm -V example
```

This command will report any file discrepancies in the package.

-->Uninstalling RPM Packages

```
rpm -e example
```

This will uninstall the package and its associated files.

-->To list all installed packages in RPM

```
rpm -qa
```

To list all installed packages on your system, use the -q (or –query) option with the -a (or –all) flag:
This will display a list of installed packages along with their names and versions.

Network Information

Basic Network Commands

ifconfig: Configures network interfaces (deprecated, replaced by `ip`).

```
sudo apt install net-tools
```

```
ubuntu@ip-172-31-32-223:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.32.223  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::24:8fff:fe4e:24b9  prefixlen 64  scopeid 0x20<link>
        ether 02:24:8f:4e:24:b9  txqueuelen 1000  (Ethernet)
        RX packets 4274  bytes 863641 (863.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4003  bytes 495118 (495.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 164  bytes 18092 (18.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 164  bytes 18092 (18.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ip: Shows/manipulates routing, devices, policy routing, and tunnels.

```
ubuntu@ip-172-31-32-223:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:24:8f:4e:24:b9 brd ff:ff:ff:ff:ff:ff
    inet 172.31.32.223/20 metric 100 brd 172.31.47.255 scope global dynamic eth0
       valid_lft 2521sec preferred_lft 2521sec
    inet6 fe80::24:8fff:fe4e:24b9/64 scope link
       valid_lft forever preferred_lft forever
```

(or) **ubuntu@ip-172-31-32-223:~$ ip link show**

ubuntu@ip-172-31-32-223:~$ ip route show

```
ubuntu@ip-172-31-32-223:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.32.223  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::24:8fff:fe4e:24b9  prefixlen 64  scopeid 0x20<link>
        ether 02:24:8f:4e:24:b9  txqueuelen 1000  (Ethernet)
        RX packets 7050  bytes 1120422 (1.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7992  bytes 901616 (901.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ping: A ping (Packet Internet or Inter-Network Groper) is a basic Internet program that allows a user to test and verify if a particular destination IP address exists and can accept requests in computer network administration.

(**or**) 1. Check network connectivity   2. Check network connection

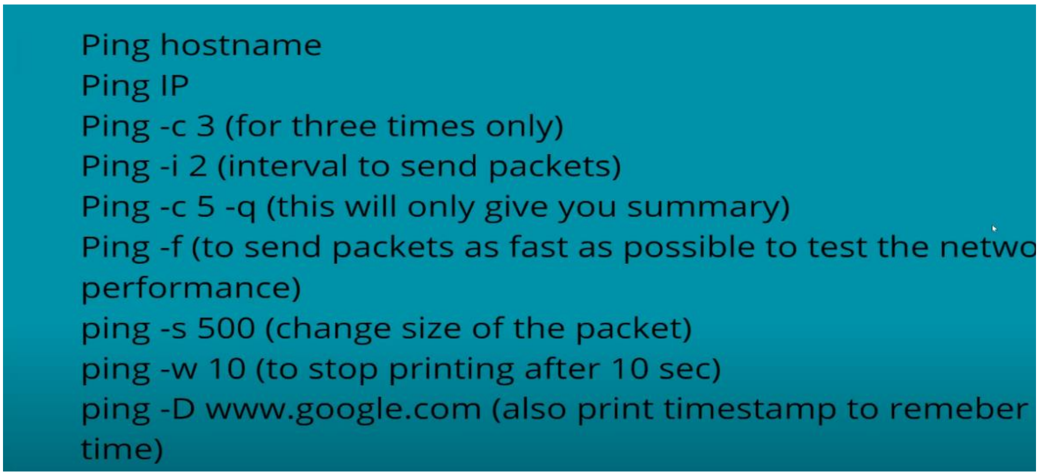
3.Check network interface card   4. Check latency on network.

5.DNS resolution

```
Ex: ubuntu@ip-172-31-32-223:~$ ping google.com

PING google.com (142.250.199.142) 56(84) bytes of data.

64 bytes from bom07s36-in-f14.1e100.net (142.250.199.142): icmp_seq=1 ttl=54 time=60.9 ms

64 bytes from bom07s36-in-f14.1e100.net (142.250.199.142): icmp_seq=2 ttl=54 time=60.9 ms

--- google.com ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms

rtt min/avg/max/mdev = 60.853/60.880/60.907/0.027 ms
```

ICMP – (Internet Control Message Protocol)

```
ubuntu@ip-172-31-32-223: ~$ ping -c5 google.com
```



Ping hostname
Ping IP
Ping -c 3 (for three times only)
Ping -i 2 (interval to send packets)
Ping -c 5 -q (this will only give you summary)
Ping -f (to send packets as fast as possible to test the netwo
performance)
ping -s 500 (change size of the packet)
ping -w 10 (to stop printing after 10 sec)
ping -D www.google.com (also print timestamp to remeber
time)

netstat:

netstat is a command line utility.

a) Network connections for TCP OR YDO

b) ROUTING TABLES

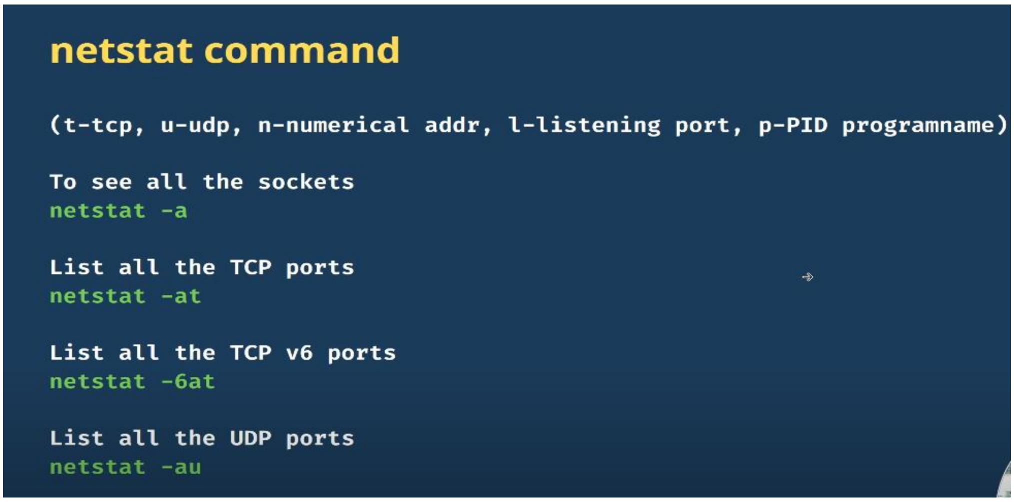
c)A, number of network work interface

d)Network protocol sta

→ case:

To identify number of connections on a given port or ip.

netstat -putan |grep :22



netstat command

(t-tcp, u-udp, n-numerical addr, l-listening port, p-PID programname)

To see all the sockets
netstat -a

List all the TCP ports
netstat -at

List all the TCP v6 ports
netstat -6at

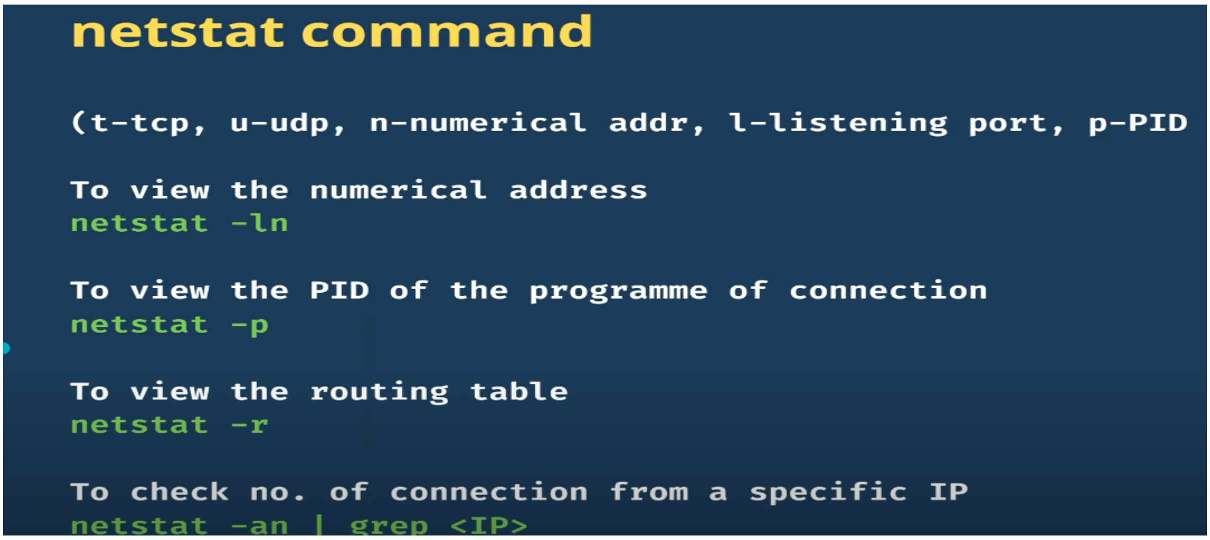
List all the UDP ports
netstat -au

List all listening ports

netstat -l

it will list all in numerical format

ubuntu@ip-172-31-32-223: ~$ netstat -atln



netstat command

(t-tcp, u-udp, n-numerical addr, l-listening port, p-PID

To view the numerical address
netstat -ln

To view the PID of the programme of connection
netstat -p

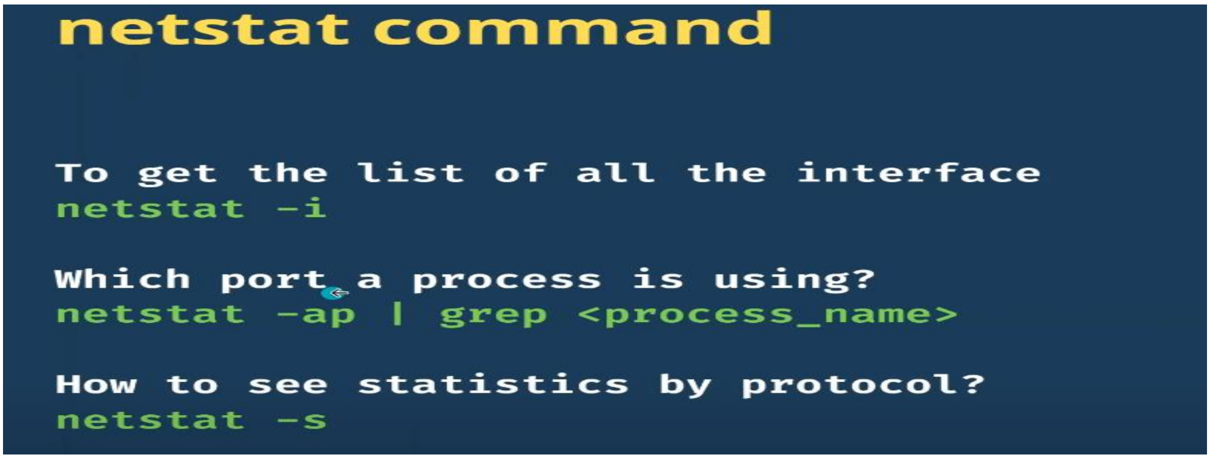
To view the routing table
netstat -r

To check no. of connection from a specific IP
netstat -an | grep <IP>

```
ubuntu@ip-172-31-32-223:~$ sudo netstat -ap|grep apache2
tcp6       0      0 [::]:http               [::]:*                  LISTEN
12604/apache2
```

```
ubuntu@ip-172-31-32-223:~$ sudo netstat -au | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

```
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 ip-172-31-32-223:bootpc 0.0.0.0:*
udp        0      0 localhost:323           0.0.0.0:*
udp6       0      0 ip6-localhost:323       [::]:*
```



netstat command

To get the list of all the interface
`netstat -i`

Which port a process is using?
`netstat -ap | grep <process_name>`

How to see statistics by protocol?
`netstat -s`

These was all about the netstat command. For further information you can watch these → (https://www.youtube.com/watch?v=YuTfG-mVRlA&t=621s)

ss:

The ss (socket statistics) command in Linux is a powerful tool for examining network connections and socket statistics. It provides more detailed information than older tools like netstat, and it is faster and more efficient.

a) Monitoring Server Connections

```
ubuntu@ip-172-31-32-223:~$ ss -t -a
```

This command lists all active TCP connections. You can look for connections to the web server's port (typically 80 for HTTP or 443 for HTTPS).

b) Checking Listening Ports

```
ubuntu@ip-172-31-32-223:~$ ss -lt
```

This command shows all listening TCP ports. You can check if the service is listening on the expected port (e.g., port 80 for Apache).

c) Troubleshooting Slow Connections

Identify established connections and check if there are any delays or issues with connections.

```
ubuntu@ip-172-31-32-223:~$ ss -t -o state established
```

This command lists all established TCP connections. The `-o` option provides additional information such as timers, which can help diagnose issues like timeouts.

d) Identifying High Network Usage by Processes

```
ubuntu@ip-172-31-32-223:~$ ss -p
```

This command shows all sockets along with the associated process using them. This can help identify if a specific process is causing high network usage.

e) Filtering Connections by Port

Filter and display all connections to a specific port

```
ubuntu@ip-172-31-32-223:~$ ss -at 'dport = :3306'
```

This command filters all TCP connections with the destination port 3306

f) Summary of Socket Statistics

Get a summary of the current socket usage statistics to monitor overall network health.

```
ubuntu@ip-172-31-32-223:~$ ss -s
```

This command provides a summary of socket statistics, including the number of TCP, UDP, and other socket types. It helps in quickly assessing the network's current state.

g) Utility to investigate sockets

```
ubuntu@ip-172-31-32-223:~$ ss -tuln
```

-t: TCP sockets.

-u: UDP sockets.

-l: Listening sockets.

-n: Do not resolve names.

traceroute:

Tracks the route packets on how the data travels on internet from your computer to destination.

The only required parameter is the name or IP address of the destination host.

How traceroute Works

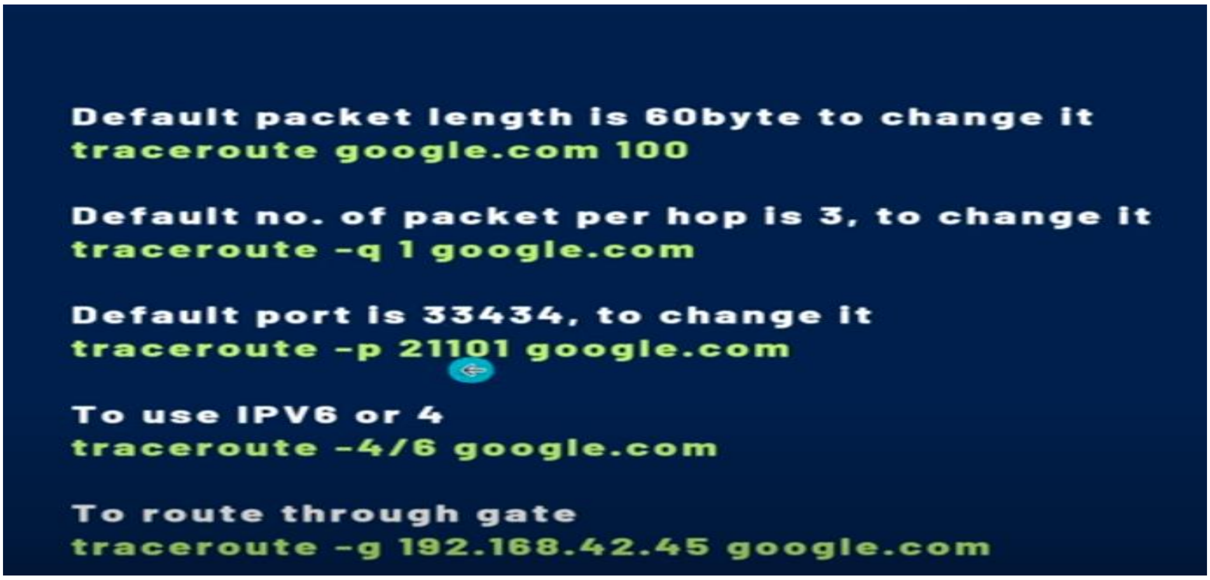
When you run traceroute, it sends packets with gradually increasing Time-To-Live (TTL) values. The TTL value determines the number of hops (routers) a packet can traverse before being discarded. When a router discards a packet due to the TTL reaching zero, it sends back an ICMP "Time Exceeded" message, which traceroute uses to identify the router. By starting with a TTL of 1 and incrementing by 1 for each set of packets, traceroute maps out the path to the destination.

a) Basic Syntax:

ubuntu@ip-172-31-32-223: ~$ traceroute google.com

b) -n: Display numerical IP addresses instead of trying to resolve hostnames.

ubuntu@ip-172-31-32-223:~$ traceroute -n google.com



```
Default packet length is 60byte to change it
traceroute google.com 100

Default no. of packet per hop is 3, to change it
traceroute -q 1 google.com

Default port is 33434, to change it
traceroute -p 21101 google.com

To use IPV6 or 4
traceroute -4/6 google.com

To route through gate
traceroute -g 192.168.42.45 google.com
```

c) -m: Set the maximum number of hops.

ubuntu@ip-172-31-32-223: ~$ traceroute -m 50 google.com

d) -I: Use ICMP ECHO instead of UDP datagrams.

ubuntu@ip-172-31-32-223: ~$ traceroute -I google.com

e)-T: Use TCP SYN packets instead of UDP datagrams.

ubuntu@ip-172-31-32-223: ~$ traceroute -T google.com

route: Show or manipulate the IP routing table.

ubuntu@ip-172-31-32-223:~$ route -n

Kernel IP routing table

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.31.32.1     0.0.0.0         UG    100    0        0 eth0
172.31.0.2      172.31.32.1     255.255.255.255 UGH   100    0        0 eth0
172.31.32.0     0.0.0.0         255.255.240.0   U     100    0        0 eth0
172.31.32.1     0.0.0.0         255.255.255.255 UH    100    0        0 eth0
```

hostname: Display or set the system's hostname.

```
ubuntu@ip-172-31-32-223: ~$ hostname
ip-172-31-32-223
ubuntu@ip-172-31-32-223:~$ hostname -I
172.31.32.223
```

Network Management Commands

nmcli: Command-line tool for managing NetworkManager.

```
ubuntu@ip-172-31-32-223:~$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  unmanaged  --
lo      loopback  unmanaged  --
ubuntu@ip-172-31-32-223:~$ nmcli connection show
```

nmtui: Text user interface for NetworkManager.

```
ubuntu@ip-172-31-32-223:~$ nmtui
```

DNS Commands

nslookup: The `nslookup` command is a network administration tool used for querying the Domain Name System (DNS) to obtain domain name or IP address mapping information. It's commonly used to troubleshoot DNS-related issues, check DNS records, and diagnose network problems

```
ubuntu@ip-172-31-32-223:~$ nslookup cinystore.com
Server:         127.0.0.53
Address:        127.0.0.53#53
```

```
Non-authoritative answer:
Name:   cinystore.com
Address: 20.192.170.15
```

dig: dig command stands for **Domain Information Groper**. It is used for retrieving information about DNS name servers. It is basically used by network administrators. It is used for verifying and troubleshooting DNS problems and to perform DNS lookups.

Installing Dig command

In case of Debian/Ubuntu

```
$sudo apt-get install dnsutils
```

In case of CentOS/RedHat

```
$sudo yum install bind-utils
```

```
ubuntu@ip-172-31-32-223:~$ dig www.cinystore.com
```

b) To query domain "A" record with +**short**

```
ubuntu@ip-172-31-32-223:~$ dig cinystore.com +short

20.192.170.15
```

c) To query all DNS record types. We use "ANY" option to query all the available DNS record types associated with a domain. It will include all the available record types in the output.

```
ubuntu@ip-172-31-32-223:~$ dig www.cinystore.com ANY
```

d)To query MX record for the domain.

```
ubuntu@ip-172-31-32-223:~$ dig www.cinystore.com  MX
```

e) To trace DNS path +trace" command is used for tracing the DNS lookup path. This option makes iterative queries to resolve the name lookup. It will query the name servers starting from the root and subsequently traverses down the namespace tree using iterative queries following referrals along the way.

```
ubuntu@ip-172-31-32-223:~$ dig www.cinystore.com +trace
```

host: The 'host' command in Linux is primarily used for performing DNS lookups and reverse lookups. This means it can translate domain names into IP addresses and vice versa. Let's break down these two primary functions.

a) Display Domain's IP Address

```
ubuntu@ip-172-31-32-223: ~$ host www.cinystore.com
```

b) Use only Ipv4 for Query Transport

```
ubuntu@ip-172-31-32-223:~$ host -4 www.cinystore.com
```

c) Use only Ipv6 for Query Transport

```
ubuntu@ip-172-31-32-223:~$ host -6 www.cinystore.com

;; communications error to ::1#53: connection refused

;; communications error to ::1#53: connection refused

;; no servers could be reached
```

d) Display Domain Name Servers

```
ubuntu@ip-172-31-32-223:~$ host -t ns www.cinystore.com
```

e) Display only Cname Records

```
ubuntu@ip-172-31-32-223:~$ host -t cname www.cinystore.com
```

f) Display only MX Records

```
ubuntu@ip-172-31-32-223:~$ host -t mx www.cinystore.com
```

g) Display only TXT Records

```
ubuntu@ip-172-31-32-223:~$ host -t txt www.cinystore.com
```

Network Configuration and Services

ethtool: Display or change Ethernet device settings.

```
ubuntu@ip-172-31-32-223:~$ sudo ethtool eth0

Settings for eth0:

        Link detected: yes
```

{This lists all network interfaces by their names.}

```
ls /sys/class/net/
```

iwconfig: Configure wireless network interfaces.

Installation -- > **sudo apt install wireless-tools**

```
ubuntu@ip-172-31-32-223:~$ iwconfig
```

O/P:

```
lo        no wireless extensions.
eth0      no wireless extensions.
```

Here lo, eth0 are network interfaces available on system.

ip link set: Bring an interface up or down.

Using the command **{**ip link set eth0 down**}** disables the network interface eth0, which can lead to loss of network connectivity on that interface. If eth0 is the primary network interface for your instance, this would disconnect it from the network, rendering it inaccessible remotely.

```
ubuntu@ip-172-31-32-223:~$   ip link set eth0 down
```

After giving these command instance will not work.

How to reconnect again**?**

First, reboot the instance or create an alarm until it will get 2/2 check passed.

Bring Interface Back Up

```
ubuntu@ip-172-31-32-223:~$ ip link set eth0 up
```

iptables: The iptables command in Linux is a powerful tool that is used for managing the firewall rules and network traffic. It facilitates allowing the administrators to configure rules that help how packets are filtered, translated, or forwarded. On using this iptables, you can set up security policies to control incoming and outgoing traffic, define port forwarding, and implement network address translation (NAT).

a) ubuntu@ip-172-31-32-223:~$ sudo iptables -L

O/P:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

b) ubuntu@ip-172-31-32-223:~$ iptables -I INPUT -s 10.0.0.1 -j DROP

c)For example I must block security group in instance

```
  iptables -I INPUT -p tcp  --dport 80 -j DROP
```

d)For example I must restart again

```
  iptables -I INPUT -p tcp  --dport 80  -s 27.59.55.113 -j ACCEPT
```

https://www.whatismyip.com/ip-address-lookup/

Network Monitoring and Debugging

tcpdump: Network packet analyzer.

tcpdump -i eth0

iftop: Display bandwidth usage on an interface by host.

iftop -i eth0

nload: Display network usage.

sudo apt install nload

nload

iperf3: Network bandwidth measurement tool.

iperf3 -s (start server)

iperf3 -c 13.201.117.167 (start client)

mtr: Network diagnostic tool that combines ping and traceroute.

```
mtr google.com
```

System Maintenance and Backup File Compression

gzip

gzip stands for GNU zip. It is a file compression utility used to reduce the size of files, making them easier to store and faster to transfer. It uses the DEFLATE compression algorithm, which is a combination of LZ77 and Huffman coding.

a) Compress a Single File:

```
ubuntu@ip-172-31-32-223:~$ gzip file3
```

O/P: file3.gz

b) Compress Multiple Files Individually:

```
ubuntu@ip-172-31-32-223:~$ gzip file1 file2 file3 file4
ubuntu@ip-172-31-32-223:~$ ls
```

O/P:

```
file1.gz file2.gz file3.gz file4.gz
```

c) Compress All Files in a Directory:

```
gzip /home/ubuntu/project_sasi/*
```

O/P: sasi.gz

gunzip:

gunzip is a utility that decompresses files that were compressed using gzip. It essentially reverses the compression done by gzip, restoring the files to their original state.

a) Decompress a single file

```
gunzip file3.gz
```

b) Decompress multiple files

```
gunzip file1.gz file2.gz file3.gz
```

c)Decompress all files in a directory

```
gunzip /home/ubuntu/project_sasi/*.gz
```

tar:

The tar command in Linux is used to create, maintain, modify, and extract files from an archive file, typically referred to as a tarball. Here are some common examples of how to use the tar command.

Archive File:

A new file named archive.tar is created in the current directory.

```
ubuntu@ip-172-31-32-223:~$ tar cvf archive.tar file9
ubuntu@ip-172-31-32-223:~$ ls
archive.tar    file9
```

*For example, by mistake file9 has been deleted how to retrieve it .

Extracting a Single File from a tar Archive

```
ubuntu@ip-172-31-32-223:~$ tar -xvf archive.tar file9
```

Verifying Extraction

```
ubuntu@ip-172-31-32-223:~$ ls -l file9
```

By running tar -xvf archive.tar file9, you can restore file3 from archive.tar

zip: zip is a command-line utility that allows you to compress files and directories into a single archive file with the .zip extension. This helps in reducing the file size and bundling multiple files together for easier distribution and storage.

a) Create a ZIP Archive

```
ubuntu@ip-172-31-32-223:~$ zip archive.zip file1
```

O/P:

```
ubuntu@ip-172-31-32-223:~$ ls
archive.zip    file1
```

b) Add Multiple Files to a ZIP Archive

```
ubuntu@ip-172-31-32-223:~$ zip archive.zip file1 file2 file3 file4
```

O/P:

This command will create archive.zip containing file1, file2, file3, file4

c) Compress an Entire Directory

ubuntu@ip-172-31-32-223:~$ **zip -r archive.zip myfolder**

-r: Recursively compress files in directories.

myfolder: The directory to be included in the archive.

d) Update a ZIP Archive

ubuntu@ip-172-31-32-223:~$ **zip archive.zip file5**

This command will add file5 to the existing archive.zip.

unzip: unzip is a command-line utility that allows you to extract files and directories from a ZIP archive. It can decompress files that were compressed using the zip utility or any other program that produces ZIP files.

a) Extract a ZIP archive:

ubuntu@ip-172-31-32-223:~$ **unzip archive.zip**

b) Extract to a specific directory:

unzip archive.zip -d /home/ubuntu/Prasanth

This command extracts the contents of archive.zip into the specified destination directory. (/home/ubuntu/Prasanth)

c)List the contents of a ZIP archive**:**

ubuntu@ip-172-31-32-223: ~$ **unzip -l archive.zip**

Security

SSH Access

Logging in to the Remote Server

To log in to the remote server using the ubuntu user:

→ ssh ubuntu@13.201.117.167

For accessing above first, we must do these

cd /home/ubuntu/.ssh

On Your Local Machine, display the public key:

```
cat ~/.ssh/id_rsa.pub
```

➔Copy the Public Key: Copy the entire content of the id_rsa.pub file.

On the Remote Server, verify that the key is correctly added

```
cat ~/.ssh/authorized_keys
```

(or) **Append the Public Key in remote server.**

```
echo "ssh-rsa AAAAB3Nza... your_public_key" >> ~/.ssh/authorized_keys
```

--- Configuring SSH Access for ubuntu---

```
sudo apt-get install openssh-server   # Ubuntu
sudo systemctl start ssh
sudo systemctl enable ssh
```

SCP Usage

Copying a File to the Remote Server

To copy a file from your local machine to the remote server as the ubuntu user:

Example: scp /home/youruser/file.txt ubuntu@13.201.117.167:/home/ubuntu/

Copying a File from the Remote Server

Example:

```
scp ubuntu@13.201.117.167:/home/ubuntu/file.txt /home/youruser/
```

Copying a Directory to the Remote Server

To copy a directory from your local machine to the remote server:

```
scp -r /home/youruser/localdir ubuntu@13.201.117.167:/home/ubuntu/
```

Copying a Directory from the Remote Server

To copy a directory from the remote server to your local machine:

```
scp -r ubuntu@13.201.117.167:/home/ubuntu/remotedir    /home/youruser/
```

Downloading files from the web and performing HTTP requests

wget

wget is a command-line utility for downloading files from the web. It supports HTTP, HTTPS, and FTP protocols.

a) Downloading a file

Download a single file from a URL:

```
wget https://www.cinystore.com/
```

This command will download the homepage of https://www.cinystore.com/ and save it as index.html

b) Download and Rename the File

```
wget -O cinystore_homepage.html https://www.cinystore.com/
```

This will download the homepage and save it as cinystore_homepage.html

c) Download Entire Website

To download the entire website recursively, you can use the following command:

```
wget -r -l 5 https://www.cinystore.com/
```

Download the Content:

```
wget --recursive --no-parent --convert-links --adjust-extension --page-requisites --no-clobber https://www.cinystore.com/corporate/
```

Create a Zip File:

```
zip -r cinystore_corporate.zip cinystore.com/corporate
```

***** By following these steps, you will be able to download the content from https://cinystore.com/corporate/ and create a zip file named cinystore_corporate.zip containing the downloaded content.

curl

curl is a command-line tool for transferring data with URLs. It supports various protocols, including HTTP, HTTPS, FTP, and more. It is also capable of performing more complex operations like making API requests.

a) Download the Homepage

```
curl -O https://www.cinystore.com/
```

b) Download and Rename the File

curl -o cinystore_homepage.html https://www.cinystore.com/

This will download the homepage and save it as cinystore_homepage.html

c) **Make a GET Request**

curl https://www.cinystore.com/

You can use curl to make a simple GET request to retrieve the homepage and display it in the terminal:

d) **Follow Redirects**

If the URL redirects to another URL, you can use the -L option to follow redirects:

curl -L -O https://www.cinystore.com/

e) **Save HTTP Response Headers**

To save the HTTP response headers to a file while downloading the content:

curl -D headers.txt -o cinystore_homepage.html https://www.cinystore.com/

f) **Header of the website**

curl -I https://www.cinystore.com

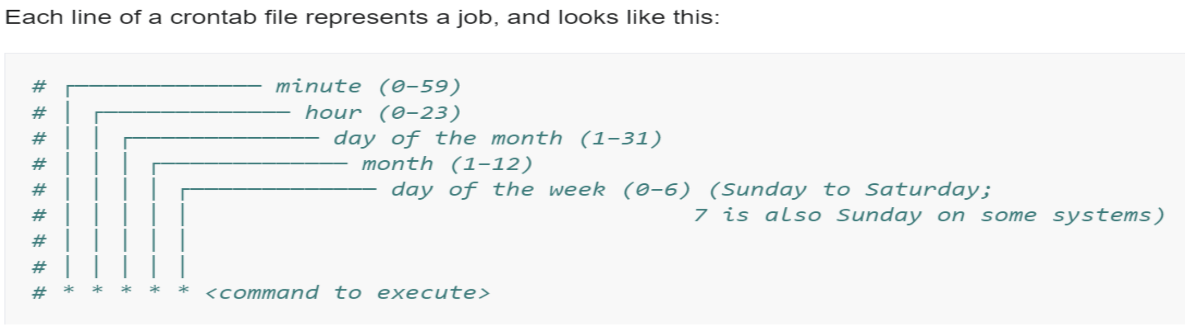
If website is up or not, if it is 200 Ok website is Up.

System Maintenance

Job Scheduling and Task Automation

cron

The cron command-line utility is a job scheduler on Unix-like operating systems. Users who set up and maintain software environments use cron to schedule jobs (commands or shell scripts), also known as cron jobs, to run periodically at fixed times, dates, or intervals.



Each line of a crontab file represents a job, and looks like this:

```
# ┌───────────── minute (0–59)
# │ ┌───────────── hour (0–23)
# │ │ ┌───────────── day of the month (1–31)
# │ │ │ ┌───────────── month (1–12)
# │ │ │ │ ┌───────────── day of the week (0–6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

a) For example, I created a Notify_SMS.sh and I had given a chmod 755 Notify_SMS.sh

https://www.geeksforgeeks.org/cron-command-in-linux-with-examples/

b) Write the script & add the following content to Notify_SMS.sh by using command given below

sudo nano Notify_SMS.sh

c) After that Open the crontab file:

crontab -e

d) Add a cron job to run a script every day at 5:00 PM:

00 17 * * * /home/ubuntu/Notify_SMS.sh

e) List Cron Jobs**:**

crontab -l

f) Remove Crontab**:**

crontab -r

at: The at command in Linux is used to schedule a command or script to be executed at a specified time in the future. It allows users to run commands once at a specific time without manual intervention.

1.Create and Make the Script Executable:

a) For example, I created a Notify_SMS.sh and I had given a chmod 755 Notify_SMS.sh

https://www.geeksforgeeks.org/cron-command-in-linux-with-examples/

b) Write the script & add the following content to Notify_SMS.sh by using command given below

sudo nano Notify_SMS.sh

c) Make the Script Executable:

chmod +x /home/ubuntu/ Notify_SMS.sh

2. Schedule the Script with at

To schedule the script to run at 5:00 PM (17:00), use the following command:

at 17:00

Then, enter the command to run the script and press Ctrl+D

at> /home/ubuntu/ Notify_SMS.sh

at> <Ctrl+D>

3. Viewing Scheduled at Jobs (Verify the Job)

atq

(or) **at -l**

4. Remove a Scheduled Job:

```
atrm 143
```

(or)

```
at -r 143
```

at -r or atrm command is used to deletes job. Here 143 is the job number.

System and Service Management

Shutting Down the System

init 0: This command brings the system down to runlevel 0, which is the halt state. This essentially shuts down the system.

Immediate Shutdown: It will immediately shutdown the system.

```
ubuntu@ip-172-31-7-248:~$ sudo init 0
```

Schedule Shutdown: Schedule the system to shut down in 5 minutes.

```
ubuntu@ip-172-31-7-248:~$ sudo shutdown -h +5
```

(**or**) Alternate command given below for init 0

```
ubuntu@ip-172-31-7-248:~$ sudo systemctl poweroff
```

Rebooting the System

init 6: This command brings the system to runlevel 6, which is the reboot state. This restarts the system.

Immediate Reboot: If an immediate reboot is required.

```
ubuntu@ip-172-31-7-248:~$ sudo init 6
```

Schedule Reboot: Schedule the system to reboot in 5 minutes.

```
ubuntu@ip-172-31-7-248:~$ sudo shutdown -r +5
```

(**or**) Alternate command given below for init 6

```
ubuntu@ip-172-31-7-248:~$ sudo systemctl reboot
```

systemctl

systemctl is the command used to interact with the systemd system and service manager. systemd is a modern init system used by many Linux distributions to bootstrap the user space and manage system processes.

Here I’m taking webserver **nginx** as an example.

a) **Start a Service**

```
systemctl start nginx
```

b) **Stop a Service**

```
systemctl stop nginx
```

c) **Restart a Service**

```
systemctl restart nginx
```

d) **Enable a Service**

```
systemctl enable nginx
```

e) **Disable a Service**

```
systemctl disable nginx
```

f) **Check the Status of a Service**

```
systemctl status nginx
```

g) **Reload Service Configuration**

```
systemctl reload nginx
```

service

The service command is used to run a System V init script in as predictable an environment as possible, removing most environment variables and setting the current working directory to /. It provides a backward-compatible interface to start, stop, and check the status of services.

Here I’m taking webserver **nginx** as an example.

a) **Start a Service**

```
service nginx start
```

b) **Stop a Service**

```
service nginx stop
```

c) **Restart a Service**

```
service nginx restart
```

d) **Check the Status of a Service**

service nginx status

System Logs

dmesg

The dmesg command in Linux is used to display messages from the kernel ring buffer. These messages can provide valuable insights into system boot processes, hardware initialization, and other kernel-related events. It is particularly useful for diagnosing hardware and driver issues.

a) Viewing Kernel Messages

To display all kernel messages:

dmesg

Usage: This command outputs a list of kernel messages since the last boot. These

messages include information about hardware detection, driver initialization, and other kernel events.

b) Filtering Specific Messages

To filter and display messages related to USB devices:

dmesg | grep usb

Usage: This command pipes the output of dmesg to grep to show only messages containing the word "usb". It is useful for diagnosing USB device issues.

c) Real-Time Monitoring

To continuously monitor kernel messages in real-time:

dmesg -w

Usage: The -w (or --follow) option allows you to watch the kernel ring buffer as new messages are added. This is useful for monitoring system events as they happen.

d) Viewing Messages in Human-Readable Format

To display kernel messages with human-readable timestamps:

dmesg -T

Usage: The -T option converts the timestamps in the kernel messages to a human-readable format, making it easier to correlate events with real-world time.

e) Saving Output to a File

To save the output of dmesg to a file for later analysis:

```
dmesg > /home/ubuntu/Janu.txt
```

Usage: This command redirects the output of dmesg to a specified file. This is useful for sharing logs or performing detailed analysis later.

journalctl

The journalctl command in Linux is used to query and display logs from the systemd journal. Systemd is a system and service manager for Linux operating systems, and it maintains a binary log file for system messages. journalctl provides a powerful way to access and filter these logs.

a) Viewing All Logs

To display all logs:

```
journalctl
```

Usage: This command outputs all available journal logs, starting from the oldest entry.

It's useful for a comprehensive overview of system events.

b) Viewing Logs Since Boot

To display logs from the current boot:

```
journalctl -b
```

Usage: The -b option shows logs from the current boot. This is useful for diagnosing issues that occurred since the system last started.

c) Filtering Logs by Time

*To display logs from the last 30 minutes:

```
journalctl --since "30 minutes ago"
```

Usage: This command shows logs from the last 30 minutes, which can be helpful for recent troubleshooting.

*To display logs from a specific date and time:

```
journalctl --since "2023-07-21 12:00:00" --until "2023-07-21 13:00:00"
```

Usage: This command shows logs between the specified times. Adjust the dates and times as needed for your specific situation.

d) Filtering Logs by Service

To display logs for a specific service, such as nginx:

```
journalctl -u nginx
```

Usage: The -u option followed by the service name filters the logs to show only those entries related to the specified service. This is useful for debugging service-specific issues.

e) Real-Time Log Monitoring

To monitor logs in real-time:

```
journalctl -f
```

Usage: The -f option (follow) continuously displays new log entries as they are written to the journal. This is useful for real-time monitoring of system events.

f) Viewing Logs by Priority

To display only critical log entries:

```
journalctl -p crit
```

Usage: The -p option followed by a priority level filters logs by the specified priority. Valid priorities include emerg, alert, crit, err, warning, notice, info, and debug.

g) Saving Logs to a File

```
journalctl > /home/ubuntu/Ashwik
```

Usage: This command redirects the output of journalctl to a specified file, allowing for easier sharing and analysis.

System and Information Commands

echo

The echo command is used to display text or output the value of a variable to the terminal. It's a simple yet powerful command commonly used in shell scripting and command-line operations.

a) Displaying a Simple Message

```
echo "Hello, World!"
```

O/P:

```
Hello, World!
```

b) Displaying the Value of a Variable

```
name="Prasanth Reddy" echo "hello, $name!"
```

O/P:

```
hello, Prasanth Reddy!
```

c) Creating a File with Content

```
echo "This is a test file." > testfile
```

O/P:

This command creates a file named testfile with the content "This is a test file."

d) Appending Content to a File

```
echo "This is an additional line." >> testfile
```

O/P:

```
This is a test file.

This is an additional line.
```

e) Using Escape Sequences

```
echo -e "Hi Iam Prasanth\nCompleted AWS DevOps\nPresent at Hyderabad "
```

The -e option enables the interpretation of escape sequences, allowing \n to create new lines.

O/P:

```
Hi Iam Prasanth

Completed AWS

Present at Hyderabad
```

f) Suppressing the Trailing Newline

```
echo -n "This is a single line without a newline at the end."
```

The -n option suppresses the trailing newline.

O/P:

```
This is a single line without a newline at the end.
```

date

The date command is used to display or set the system date and time. It can also be used to format the date and time in various ways. This command is essential for tasks that involve scheduling, logging, and time-based operations.

a) Displaying the Current Date and Time

```
date
```

O/P:

```
Mon Jul 22 05:27:28 UTC 2024
```

b) Display in Date in Indian Time

```
TZ=Asia/Kolkata date
```

O/P:

```
Mon Jul 22 10:58:03 IST 2024
```

c) Displaying the Date in UTC

```
date -u
```

O/P:

```
Mon Jul 22 06:09:26 UTC 2024
```

d) Displaying Date in a Custom Format

```
date +"%Y-%m-%d %H:%M:%S"
```

This command formats the date and time in YYYY-MM-DD HH:MM:SS format.

O/P:

```
2024-07-22 05:44:37
```

e) Adding or Subtracting Days from the Current Date

```
   date --date="2 days ago"
```

This command will print the date as 2 days ago as per the given date

```
  date --date="2 days hence"
```

This command will print the date as 2 days after as per the given date

timedatectl

The timedatectl command is used to query and change the system clock and its settings.

a) List Available Time Zones

```
timedatectl list-timezones
```

This command lists all available time zones.

```
timedatectl list-timezones | grep Asia/Kolkata
```

You can scroll through the list or use grep to find a specific time zone.

b) Set the Time Zone:

```
sudo timedatectl set-timezone Asia/Kolkata
```

This command sets the system time zone to Asia/Kolkata.

c) Verify the Change:

```
timedatectl
```

cal

The cal command in Linux is used to display a calendar in the terminal. It is a simple yet useful command for quickly viewing a calendar for a specific month or year. The cal command can display the current month's calendar, a specific month, or an entire year.

a) Displaying the Current Month's Calendar

```
cal
```

This command displays the calendar for the current month.

b) Displaying a Specific Month's Calendar

```
cal 12 2024
```

This command displays the calendar for December 2024.

c) Displaying the Entire Year Calendar

```
cal 2024
```

This command displays the calendar for the entire year 2024.

d) Displaying the Previous, Current, and Next Month

```
cal -3
```

This command displays the previous, current, and next month's calendars.

e) I had to specify certain date in a month

```
cal 7 2024 | grep -C2 '15'
```

f) I had to specify number of dates in a month

```
cal 7 2024 | grep -C2 -E '15|20|'
```

history

The history command in Linux is used to display a list of previously executed commands in the current terminal session. This is particularly useful for recalling commands, re-executing them, or identifying the sequence of operations performed.

a) Displaying Command History

```
history
```

b) Re-executing a Command

```
!2001
```

You can re-execute a command from the history by referencing its number. For example, to re-execute command number 2001

c) Searching Command History

```
history | grep cd
```

d) Clearing Command History

```
history -c
```

This command clears the current session's history.

who

The who command in Linux is used to display information about the users currently logged into the system. It provides details such as usernames, terminal names, login times, and sometimes the originating host of the user.

a) Displaying Logged-In Users

```
who
```

Execute who to see all logged-in users.

b) Displaying User Information with Detailed

```
who -a
```

Use who -a to get more detailed information, including login processes and originating IP addresses.

c) Displaying Only Usernames and TTYs

```
who -q
```

Use who -q to get a quick list of usernames and the total number of logged-in users.

O/P:

```
ubuntu
# users=1
```

whoami

The whoami command in Linux is used to display the username of the currently logged-in user. It's a simple command that helps you quickly find out which user account you are operating under at any given time.

a) Checking the Current User

```
whoami
```

b) Verifying User After Switching Accounts

```
su - hanuman
whoami
```

O/P:

```
hanuman
```

which

The which command in Linux is used to locate the path of an executable file associated with a given command. It helps identify where a command or executable is located on the filesystem, which is useful for troubleshooting and verifying that the correct version of a command is being used.

a) Finding the Path of a Command

```
which python
```

O/P:

/usr/bin/python

b) Verification:

After locating the executable, use other commands (like --version) to verify its details.

Example: python –version

info

The info command in Linux is used to display and navigate documentation for software packages and commands, typically in the GNU info format. It provides detailed information and documentation that can be more extensive than man pages.

a) Searching for Topics within info

You can search for specific topics within the info documentation. For example, to find information about "regex"

info grep

b) Searching and Navigating

When viewing info documentation, you can navigate between different sections and nodes. For example, within the info page for ls, you can press n to go to the next node or p to go to the previous node.

c) Quitting info

To exit the info interface:

*Press q to quit and return to the terminal.

sleep: The sleep command in Linux is used to pause the execution of a script or command for a specified amount of time. It can be useful in various scenarios such as creating delays in scripts, scheduling tasks, or waiting for a resource to become available.

a) Pausing for a Few Seconds

sleep 5

b) Pausing for a Specific Duration in Minutes

```
sleep 2m
```

c) Pausing for Hours

```
sleep 1h
```

d) Pausing for Days

```
sleep 1d
```

e) Using sleep in a Script

In a shell script, you might want to add a delay between commands. For example, if you want to wait 10 seconds before sending a notification:

```
#!/bin/bash

echo "Starting process..."

sleep 10

echo "Process completed."
```

clear

The clear command in Linux is used to clear the terminal screen. This command is helpful when you want to remove all previous commands and outputs from the view to have a clean working space.

a) Using the Command

```
clear
```

Type clear and press Enter

b) After Cleaning

Terminal screen is clean, showing only the prompt.

man

The man command in Linux is used to display the manual pages (man pages) for various commands, system calls, library functions, and configuration files. It provides detailed information on how to use commands and understand their options, syntax, and functionalities.

a) Viewing the Manual for a Command

```
man ping
```

The man command is an essential tool for accessing detailed information and documentation in Linux, helping users effectively use commands and understand system functionality.