**LINUX**

* **Pre – requisite software’s:** If we want to connect Linux server for write commands. We need some pre-requisite software’s.

**1.SSH (Secure Shell) Tools:**

1. For windows, we need to install Putty or Super Putty.
2. For MacBook, we have Terminal.

**Note:**

\* If you want to connect to the server, you show know about IP address/Host name, Username, Password.

**2. FTP (File Transfer Protocol) Tools:**

1) For windows, WinSCP

2) For windows/MacBook/Linux, FileZilla

**Note:**

\* There are some files in the server. If you want to retrieve or download the files using FTP protocol.

\* If you have this FTP tool in your PC, you can upload or download files.

**LINUX FILE SYSTEM HIERARCHY**

**Introduction:**

* For example, we can call as folders in windows. Same as, we can call as Directories in the Linux. There are Some sub- directories are also in the Directories.
* In the Linux, there is a root directory/Parent Directory. For this root Directory, there are some sub-directories.

**Sub Directories:**

1. **/root:** The root directory is the top most directory and this root directory contains all its nested directories and files.
2. **/home:** This home directory contains all the user’s information. It is same as users in our windows. These Users are going to create directory in the home directory.
3. **/bin:** This binary directory contains the binary files which are used for basic system functionality. In Linux, each command has the binary file.

**Ex:** mkdir

ls

cd

1. **/sbin:** This system binary directory contains the binary files.

**Ex:** useradd

userdel

shutdown

1. **/boot:** This directory contains the files needed to boot the system, such as kernel, initial RAM disk and bootloader configuration files.
2. **/etc:** This etc directory contains the all-configuration files and scripts.

**Ex:** sudoers

sshd-config

profile

shadow

group

passwd

motd

1. **/lib:** This library directory contains the all libraries of the OS.
2. **/proc**: If you execute a command, shell script it will generate a process ID for each task. All the process

Id’s information is stored in this Directory. Along with these it stores some files are cpuinfo, meminfo.

1. **/opt:** By Default, this directory is empty. All the third-party software installation will take place in this directory. If you want to install any software, you should login as root user.
2. **/tmp:** We will use this directory as Temporary storage of files.
3. **/var:** It contains the variables information’s and also log files information in this Directory.
4. **/dev:** It contains all the external hardware device information which are connected to the server.

**BASIC COMMANDS**

**1.mkdir (Making Directory):** This command is used to create the Directory.

**Syntax:** mkdir dir\_name

* **mkdir -v dir\_name:** This command is used to create and also it displays the directory is created.
* **mkdir -p dir\_name/sub\_directory/sub\_directory1/:** This command is used to create sub-directories in that directory.

**Example:** mkdir -p or -vp DevOps/Linux/ShellScripting/Git/Nexus

* **mkdir -m 700 dir\_name:** This command is used to create a directory with customize permissions.
* **mkdir dir1 dir2 dir3:** This command is used to create multiple directories at once.

**Example:** mkdir DevOps/Linux/Shell Scripting/Git

* **mkdir “my directory”:** This command is used to create a folder with spaces.

**Example:** mkdir “my line”.

* **mkdir /home/ubuntu/dir\_name:** This command is used to create a directory in specific location.

**2.ls (List):** This command is used to list the files or directories.

**Syntax**: ls

* ls -l: Displays the list of files in long format.
* ls -a: Displays the list of hidden files
* ls -al: Displays the list of Hidden files in the long format.
* **ls -lt:** Displays the files which we create lastly in the ascending order based on time.
* ls -ltr: Displays the files which we created in the reverse order.
* ls -li: Displays the files with unique inode number.
* ls -h: Displays the file-size in the human readable format.
* ls /path/to/dir\_name: It creates a directory in that location.
* ls dir\_name: It displays the list of files in the current directory.
* ls \*.txt: Displays the list of files with the .txt

**3.pwd (Present Working Directory)**: This command shows the present directory which we are in.

**Syntax:** pwd

**4.cd(Change Directory):** It Change the directory to one directory to another directory.

**Syntax:** cd

* cd dir\_name: It changes to the specific directory.
* cd dir\_structure: It changes to the directory structure.
* cd .. : It Displays one level up directory.
* cd ../..: It Displays the two-level up Directory.
* cd ~: It takes back to the home directory.
* cd -: It displays the current working directory which we are in.
* cd “Dir Name’’: It changes to the directory which we are created with spaces.

cat: Concatenate and combining files.

* It is used to create a file.
* It is used to display the content of the file.
* It is used to combining of files and copying the content to the another file.
* It is used to append the content to the file.

Syntax:

* cat file\_name : It displays the content of file.
* cat file\_name1 file\_name2 > file\_name3 – Combining of two files and copying the content to the another new file.
* cat >> file\_name1: Appending the content to the existing file.

**5.touch:** This command is used to create empty files and also update the time stamp of existing files.

**Syntax:** touch file\_name

* **touch file1 file2 file3:** It creates multiple files at once.
* **touch existing\_file:** It update the time stamp of existing file.
* **touch -t YYYYMMDDHHMM.SS file\_name:** This command is used to modify timestamps to a specific date and time.
* **touch “file with spaces.txt”:** This command is used to create a file with spaces.

**Note:** We can open the text file using vi and you can add data into the file.

**6.rmdir (remove Directory):** This command is used to remove the empty directories.

**Syntax:** rmdir dir\_name

**=>** rmdir **-**v dir\_name: This command displays that the directory is removed.

=> rmdir dir1 dir2 dir3: This command is removing multiple directories at once.

=> rmdir /path/to/dir\_name: This command is removing the directory in absolute path.

=> rmdir -p parent/child: This is removing the parent and child directories if they are empty.

=> rmdir “my directory”: This command is used to remove the directory with spaces**.**

**7.rm(remove):** This command is used to remove files or directories or directory structure(forcely).

**Syntax:** rm -rf dir\_structure – Removes the directory structure.

* rm file\_name: Removes the file.
* rm file1 file2 file3: It removes multiple files at once.
* rm -v file\_name: It removes file and also it displays as file is removed.
* rm -d dir\_name: Removes an empty directory.
* rm -rf dir\_name: Removes directory and its content.
* rm -f file\_name: Forces the removal of a file without prompting for confirmation, even if the file is write-protected.
* rm -i file\_name: Prompts for confirmation before deleting each file.
* rm “file with spaces”: Removes a file with spaces.

**8.yum**: This command is used to install or uninstall or updating the packages.

**Syntax**: sudo yum options command package\_name

* **sudo yum install package\_name**: This command is used to Installs the specified package.
* **sudo yum update package\_name**: This command is used to updates the specified package to the latest version.
* **sudo yum update:** Updates all installed packages to their latest versions.
* **sudo yum remove package\_name:** Removes the specified package
* **sudo yum list installed:** Displays a list of all installed packages.
* **sudo yum list available:** Lists all available packages.
* **sudo yum info package\_name:** Displays detailed information about the specified package.
* **sudo yum groupinstall “group\_name”:** Installs a group of related packages.
* **sudo yum groupremove “group\_name”:** Removes a group of related packages.

**9.tree:** This command is used to displaying the directory structure in a tree-like format.

Syntax: tree dir\_structure

* tree: Displays the directory structure of the current directory.
* tree /path/to/directory: Displays the directory structure of the specified directory.
* tree -l depth: Limits the depth of the directory tree to the specified level.
* tree -a: Includes hidden files (those starting with a dot.) in the output.
* tree -h: Displays file sizes in a human-readable format.
* tree -d: Lists only directories, excluding files.
* tree -f: Displays the full path for each file and directory.
* tree -p: Displays file permissions in the output.

1. **find:** The find command in Linux is a powerful utility used to search for files and directories within a directory hierarchy based on size, type, filename etc.

**Syntax:** find path expression

* **find . -type f -empty –** Displays the empty files in the current directory.
* **find ~ type f -empty –** Displays the empty files in the home directory.
* **find / -type f -empty –** Displays the empty files in the root directory (It is visible using root user).
* **find -type f -perm 0777 –** Displays all the files which have the 777 permissions.
* **find . -type f -empty –** Displays the empty directories in the current directory.

1. **umask:** umask (user creation mask) is command and also it is a setting in Linux that the default permission for the new files or directories. It works by subtracting specific permission bits from the default permissions.

**Syntax:** name

**Default Permissions:**

**Files:** 0666 (rw-rw-rw-)

**Directories:** 0777(rwxrwxrwx)

* The umask value is subtracted from the default permissions of files or directories. After subtraction, the permissions are going to set to the newly created files or directories.

**How umask Works:**

* The umask value is a 3-digit octal number (e.g., 022,002,007). Each digit represents permissions for:

1. Owner(user)
2. Group
3. Others (World)

**Example:**

* **This is for Normal User:**

1. **For file: 0 6 6 6**

**Umask: 0 0 2 2**

**----------**

**0 6 4 4**

1. **For Directory: 0 7 7 7**

**Umask : 0 0 2 2**

**-----------**

**0 7 5 5**

* **This is for Root User:**

1. **For File: 0 6 6 6**

**Umask : 0 0 0 2**

**--------------**

**0 6 6 4**

1. **For Directory: 0 7 7 7**

**Umask : 0 0 0 2**

**------------**

**0 7 7 5**

* **umask –** Displays the umask value of the user.
* **umask 022** – This command is going to set the umask value to the specific number.
* **umask 077** – This command is used to restrict umask to ensure that newly created files and directories are only accessible by the owner.
* **umask 002** – This command is used to in the environments where group collaboration is required, use umask like 002 to allow group members to write to files or directories.
* **umask 022** – This command set a default umask for all the users by configuring it in system-wide configuration like /etc/profile or /etc/bash.bashrc.
* **umask 077** – This is used for temporarily change the umask for specific reason to our files**.**

**Note:** Based on the umask value, the default permissions are going to be set to a file or directory. We can change the umask value on various criteria. This umask value is temporary what you changed now. When you re-login, the umask will be default.

* **~ / .bash\_profile**: In this, we can add the umask value for the normal users.
* **/etc/profile:** In this**,** we can add the umask value for the all users.

**3.chmod:** This command is used to change the permissions of files and directories. Permissions control like who can read, write, or execute. chmod allows you to modify these permissions for the owner, group, others**.**

**Permissions in Linux:**

In Linux, every file or directory has three types of permissions:

1. **Read(r):** This will allows viewing the contents of file.
2. **Write(w):** This will allows modifying a file or adding files in a directory.
3. **Execute(x):** This allows executing a file or accessing a directory.

These permissions are assigned to the three categories of the users:

1. **Owner(u):** The user who owns the file or directory.
2. **Group(g):** The group that owns the file or directory.
3. **Others(o):** All other users.

**Syntax:** chmod options permissions file\_name

* **chmod u+x file.txt** – This add execute permission to the user.
* **chmod o-w file.txt** – This removes the write permission to the others.
* **chmod g=rw file.txt** – This set read and write permissions to the group.
* **chmod 644 file.txt** – This command is used to set like rw-r—r to the file.
* **chmod 700 file.txt** – Set the permissions to the file.
* **chmod** **777 DevOps/** - It is only giving all the permissions to the DevOps directory not to the sub directory.
* **chmod -R 777 DevOps/** - It is going to set all the permissions to the sub directories also
* **chmod 000 file.txt** – Removes all the permissions.
* **chmod ugo+x** – It is going to set execute permissions to all.
* **chmod ugo-rwx file.txt** – Remove all permissions
* **chmod u+wx,g+r,o+x file.txt** – This is set the specific permissions to the users.

**4.chown (Change Owner):** This command is used to change the ownership of files and directories.

**Syntax:** chown options owner file.txt

* **chown username file.txt** – It change the owner of file to the specific user.
* **chown groupname file.txt** – It change the group name.
* **chown username: groupname file.tx**t – Change both the owner and group of files.
* **chown -R username: groupname directory\_name** – Change the ownership of a directory and all its content.

**5.chgrp (Change group):** This command is used to change the group ownership of files and directories. Unlike chown, which can change both the owner and group, chgrp is specifically designed to change only the group ownership.

**Syntax:** chgrp options groupname file\_name

* **chgrp groupname file.txt:** It change the group of a file to the specific group.
* **chgrp -R groupname dir\_name:** Changes the group of a directory and all its content.

**1.cp:** This command is used to copy the files and directories from one location to another location.

**Syntax:** cp options source destination

* **cp file\_name /path/to/destination:** This command is used to copy the file to the destination.
* **cp file\_name /path/to/destination/new\_filename:** This command is used to copy the file to the destination and rename it as the new file name.
* **cp -r /path/to/directory\_structure /path/to/destination:** This command is used to copy the directory structure and all it’s content to the destination.
* **cp file1 file2 /path/to/destination:** This command is going to copy the multiple files to the destination.
* **cp file\_name dir\_name:** This command is used to copy the file to the directory.
* **cp -i file\_name /path/to/destination:** If the destination file name exists the same file name, it will be prompted before overriding.

**2.mv:** This command is used to move the files and directories from one place to another place, and it renames the file.

**Syntax:** mv options source destination

* **mv file\_name /path/to/destination:** This command is used to copy the file to the specified destination.
* **mv old\_filename new\_filename:** This command is used to rename the file name.
* **mv /path/to/source\_directory path/to/destination**: This command is used to copy the entire directory to the destination.
* **mv file\_1 file\_2 /path/to/destination:** This command is used to copy the multiple files to the destination.
* **mv -i file\_name /path/to/destination:** If the file name same exists in the destination, it will be prompted before overwriting.
* **mv -f file\_name /path/to/destination:** This will overwrite the destination file without asking prompting.

**3. file:** This command is used to display the type of file. It checks the contents of file and provides information about its formats, such as whether it is a text file, binary file, executable, archieve.

**Syntax:** file options file\_name

* **file file\_name:** This command checks the type of a file.
* **file -b file\_name:** This command displays only the type of file without the file name.
* **file archieve.zip:** This command checks the compressed file type.
* **file /bin/ls:** This command checks the type of binary executable file.
* **file file.txt file.jpg file.zip:** This command is used to checks the type of multiple files at a time.

**4. ln:** The ln command is used to create links between files. Links are pointers to files and there are two types:

**a.** **Hard Links:** Direct references to the file’s data on the disk. They share the same inode number as a original file.

**Syntax:** ln options source target

* **ln file.txt hardlink.txt:** This command creates a hard link to the file so that the original file and link file share the same data on the disk.
* **ln file.txt link1 link2 link3:** This command is used to create multiple hard links to the same file.

**b. Soft/Symbolic Link:** Shortcuts that point to the file by name. They have their own inode and can points to the files or directories.

**Syntax:** ln options source target

* **ln -s file.txt slink.txt:** This command is used to create a shortcut to the file. If you delete the original file, the soft link also will be broken.
* **ln -s /path/to/file /path/to/destination/slink\_name:** This command is used to create a link in the destination directory.
* **ln -**s **/path/to/directory /path/to/slink\_name:** This command is used to create a link to the directory.

**Differences Between Hard Links and Soft Links:**

1. **Hard Link:**

* It shares the same inode as the original file.
* The Hard Link must be on the same file system.
* It cannot create link to the directories.
* The link remains valid, even the original file is deleted.
* The link file also has same size.

1. **Soft Link:**

* The soft link has its own inode.
* It can span across filesystems.
* It can also link to the directories.
* The link becomes broken if the original file is deleted.
* The soft link file size is small.

**Text Reading/Display Commands**

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

**Syntax:**

* **echo “Hello, World!”:** This command displays the string Hello, World!
* **echo “Hello World”:** This command displays the string with space.
* **echo “This is a file” > file\_name:** This command displays the string, stored in file.
* **echo “This is another file” >> file\_name:** This command will append to a string to the end of the file.
* **echo -e “This is a line.\this is another line.”:** The command displays |n is interpreted as newline character.

1. **head:** The head command in Linux is used to display the beginning portion of a file or input stream. By default, it shows the first 10 lines of the specified files.

**Syntax:**

* **head file\_name:** This command displays the first 10 lines of the file.
* **head -**n **25 file\_name:** This command displays the first 25 lines of the specific file.
* **head -c 20 file\_name**: Displays the first 20 bytes of filename. The -c option is used to specify the number of bytes.
* **head file1.txt file2.txt:** Displays the first 10 lines of both file1.txt and file2.txt.
* **head -n 5 \*.txt:** Displays the first 5 lines of all .txt files in the current directory.

1. **tail:** The tail command in Linux is used to display the last part of a file or input stream. By default, it shows the last 10 lines of the specified files.

**Syntax:**

* **tail filename.txt:** Displays the last 10 lines of filename.txt.
* **tail -n 5 filename.txt:** Displays the last 5 lines of filename.txt. The -n option allows you to specify the number of lines.
* **tail -c 20 filename.txt**: Displays the last 20 bytes of filename.txt. The -c option is used to specify the number of bytes.
* **tail file1.txt file2.txt:** Displays the last 10 lines of both file1.txt and file2.txt.
* **tail -f /var/log/syslog**:  Displays the last 10 lines of the system log file (/var/log/syslog) and continues to monitor the file for new entries.
* **tail -n 5 \*.txt:** Displays the last 5 lines of all .txt files in the current directory.

1. **sed:** The sed command is used to display the line numbers and range of lines.

**Syntax:**

* **sed -n “108p” file\_name:** Displays the 108 line of the file. Here p specifies print of that line.
* **sed -n “108,115p” file\_name:** Displays the range from 108 to 115 in the specific file.
* **sed ‘s/old/new/’ file\_name:** It replaces the old string with the new string in the file.
* **sed ‘s/old/new/2’ file\_name:** It replaces the old string with the new string of the second occurrence of a file.
* **sed ‘s/old/new/g’ file\_name:** It will print all with the new string globally temporarily.When you view the file, it is same with the old strings.
* **sed -i ‘s/old/new/g’ file\_name:** It replaced old strings with new strings permanently. It is updating the file.
* **sed '3d' filename.txt:** Deletes the 3rd line of filename.txt.
* **sed '2,5d' filename.txt:** Deletes lines 2 through 5 of filename.txt**.**
* **sed '/pattern/d' filename.txt:** Deletes all lines containing pattern in filename.txt.
* **sed '3i\New Line' filename.txt:** Inserts New Line before the 3rd line of filename.txt.
* **sed '3a\New Line' filename.txt:** Appends New Line after the 3rd line of filename.txt.

1. **more:** This command is used to display page by page of a file. Here, use ctrl+f for forwarding to the next page of a file.

**Syntax:**

* **more file\_name:** Displays the content in the file on one screen at a time.
* **more +10 file\_name:** Displays the content from the 10 th line in a file.
* **more file1\_name file2\_name:** First, displays the content of file1 after reaching the end of the file, use :n to displaying the file2 content.
* **ls -l /usr/bin | more:** Displays the output of the ls-l /usr/bin command one screen at a time.
* **more -n file\_name:** Displays the line numbers along with the content of the file.
* **more -c file\_name:** This command is used to clears the screen before displaying of the page of a file.
* **more -5 file\_name:** Displays only 5 lines of the file.

1. **less:** This command is used to display line by line of a file to see the output interactively. Unlike more, less allows both forward and backward navigation keys.

**Syntax:**

* **less filename.txt:**Opens filename.txt for interactive viewing. You can scroll up and down using the arrow keys, Page Up, Page Down, or Space.
* **ls -l /usr/bin | less:** Displays the output of the ls -l /usr/bin command in less, allowing you to scroll.
* **less file1.txt file2.txt:** Opens file1.txt first. Use :n to move to the next file (file2.txt) **.**
* **less +50 filename.txt:** Opens filename.txt and jumps directly to line 50.
* **less -N filename.txt:** Displays line numbers alongside the content of filename.txt.
* **less +G filename.txt:** Opens filename.txt and jumps to the end of the file.

1. **pipe( | ):** The command is powerful operator used to connect the output of one command to the input of another. It allows you to chain multiple commands together, enabling complex data into in a single line.

**Syntax:** command1 | command2

* **ls -l | grep ".txt":** Lists all files in the current directory and filters the output to show only lines containing .txt using grep.
* **cat filename.txt | wc -l:** Displays the contents of filename.txt and counts the number of lines.
* **cat longfile.txt | less:** Displays the contents of longfile.txt and allows you to scroll up and down.
* **cat filename.txt | grep "error":** Displays the contents of filename.txt and searches for lines containing the word error.
* **echo "hello world" | tr 'a-z' 'A-Z':** Converts the text hello world to uppercase using the tr command.
* **cat filename.txt | sed 's/old/new/g':** Replaces all occurrences of old with new in the contents of filename.txt.

1. **grep:** The name grep stands for Global Regular Expression Print. It is widely used for filtering and analysing text data**.**

**Syntax:**

* **grep "pattern" filename.txt:** Searches for the word pattern in filename.txt and displays all matching lines.
* **grep -i "pattern" filename.txt:** Searches for pattern in filename.txt, ignoring case.
* **grep "pattern" file1.txt file2.txt:** Searches for pattern in both file1.txt and file2.txt.
* **grep -c "pattern" filename.txt:** Counts the number of lines containing pattern in filename.txt.
* **grep -v "pattern" filename.txt:** Displays all lines in filename.txt that do **not** contain pattern.
* **grep "pattern" \*.txt:** Searches for pattern in all .txt files in the current directory.

1. **uptime:** The command is used to provides the information about the how long the system has been running, the current time, the number of logged-in users, and the system load averages.

**Syntax:**

* **uptime:** Displays the system uptime.
* **uptime -p:** Displays the system uptime in a more human-readable format.
* **uptime -s:** Displays the exact date and time when the system was last booted.
* **uptime && who:**Displays the system uptime and a list of currently logged-in users.

**Load Average:** Load Average in CPU processing that indicates the average number of processes that are either:

a. Running: (actively using the CPU), or

b. Waiting:  (in the run queue, ready to run but waiting for CPU resources).

**How it is calculated:**

Load average is calculated as an exponential moving average of the system's load over three-time intervals:

* 1-minute average: Represents the recent load.
* 5-minute average: Represents the medium-term load.
* 15-minute average: Represents the long-term load.

1. **Who:** The who command in Linux is used to display information about currently logged-in users on the system.

**Syntax: who**

* who -a: Displays all available information, including login time, idle time, and process IDs.
* who -q: Displays a list of usernames and the total number of logged-in users.
* who -u: Displays the login time in a more detailed format, including idle time and process ID.
* who -H: Displays column headers for the output.
* who -m: Displays information about the current user only.
* who -d: Displays information about dead processes (terminated but still listed).

1. **whereis:** It is a quick and efficient way to find the locations of these files without searching the entire filesystem.

**Syntax:** whereis command\_name

* **whereis python:** Displays the location of python.
* **whereis -b ls:** Displays only the binary file for the ls command.

1. **Date:** The date command in Linux is used to display or set the system date and time.

**Syntax:**

* **date:** Displays the current system date and time in the default format.
* **date +"%Y-%m-%d %H: %M: %S":** Displays the date and time in the format YYYY-MM-DD HH:MM: SS.
* **date +"%Y-%m-%d":** Displays only the date in the format YYYY-MM-DD.
* **date +"%H:%M:%S":** Displays only the time in the format HH:MM:SS.
* **date +"%A":** Displays the full name of the day of the week.
* **date -d "yesterday":** Displays the date and time for yesterday.
* **date -d "tomorrow":** Displays the date and time for tomorrow.
* **sudo date -s "2023-10-02 12:34:56":** Sets the system date and time to 2023-10-02 12:34:56.
* date -r filename.txt: Displays the last modification time of filename.txt.
* **date +"%Z":** Displays the current time zone.

**Disk Management Commands:**

1. **df(disk free):** The df command in Linux is used to display the amount of disk space available on file systems. It provides information about the total space, used space, available space, and the mount point of each file system.

**Syntax: df**

* **df -h:** The -h option makes the output human-readable by converting sizes into KB, MB, or GB**.**
* **df -i:** The -i option shows inode usage instead of block usage. Inodes store metadata about files.
* **df /dev/sda1:** Displays disk space usage for a specific file system.
* **df -sh /dev/sda1:** Displays disk space usage for a specific file system summary in human-readable format.
* **df --total:** The --total option adds a summary line at the end of the output, showing the total disk space usage across all file systems**.**
* **df -k:** The -k option displays disk space usage in 1K blocks.
* **df -m:** The -m option displays disk space usage in MB.

1. **du(disk usage):** The command in Linux is used to estimate and display the amount of disk space used by files and directories.

**Syntax: du**

* **du -h:** The -h option makes the output human-readable by displaying sizes in KB, MB, or GB**.**
* **du -sh /path/to/file:** Summarizes the total disk usage of the specified directory (instead of listing every file/subdirectory) and displays the size in human readable format.
* **du -h /path/to/directory:** Displays disk usage for the specified directory and its contents**.**
* **du -b /home/user:** The -b option displays disk usage in bytes instead of blocks.
* **du -m /home/user:** The -m option displays disk usage in megabytes.
* **du -ah:** Displays disk usage for all files in the human readable format.
* **du -h ~:** Displays disk usage in home directory.
* **du -sh ~:** Displays summary in human readable format in home directory.

1. **hostname:** The hostname command in Linux is used to view or set the system's hostname**.**

**Syntax:**

* **hostname:** Displays the current hostname of the system.
* **sudo hostname new-hostname**: Changes the hostname temporarily.
* **hostname -i:** Displays hostname with IP address.
* **hostname -I:** Displays all network IP’s which are associated with the host.
* **hostname -d:** Displays the DNS Domain name.
* **sudo hostnamectl set-hostname new-hostname:** It changes to the new host name permanently.

1. **ifconfig:** The command is used to configure, manage, and query network interface parameters from the command line.

**Syntax:**

* **ifconfig:** Displays the ip address of the server.
* **ifconfig -a:** It giving all the network interfaces, subnet details etc.
* **ifconfig eth0:** Displays details for a specific network interface.
* **ifconfig eth0 up:** Enables the network interface.
* **ifconfig eth0 down:** Disables the network interface.
* **ifconfig eth0 192.168.1.100:** Assigning of the static ip address to the interface.
* **ifconfig eth0 netmask 255.255.255.0:** Assigning the subnet mask to the interface.
* **ifconfig eth0 192.168.1.100 netmask 255.255.255.0:** Assigning of IP address and the netmask in a single command.

**Process Management Commands:**

1. **ps:** The ps command in Linux is used to display information about active processes running on the system.

**Syntax:**

* **ps:** Displays the processes which are associated with the current terminal.
* **ps -ef:** It gives all the information like Process ID,Name etc.
* **ps ux:** Displays the all the processes of the current user.
* **ps aux –forest:** Displays the processes in a tree format.
* **Ps -u username:** Displays the processes of the specific user.
* **Ps -p PID:** Displays the processs of the specific PID.
* **Ps -ef | grep java:** Displays the java process in all the processes.
* **Ps -ef | grep Jenkins:** Displays the Jenkins process among all the processes.

1. **kill:** The kill command is used to allow to terminate,pause or control the behavior of the process.

**Syntax:**

* **kill PID:** Kills the specific process PID.
* **kill -9 PID:** Forcefully kill the specific process PID.
* **kill PID1 PID2 PID3:** Kill multiple processes at once using PID.
* **pkill process\_name:** Kill the process specified by the process name.
* **kill -i:** Displays the all-available signals with their numbers.
* **kill -SIGNAL PID:** It sends specific signal to the PID.

1. **top:** The is command is used to provides the real-time information about the processes, usage and the system’s performance.

**Syntax:**

* **top:** It displays the real-time system process and resource usages.
* **top -u username:** Displays the processes with the specified username.

**User and Group Administrative Commands:**

1. **useradd:** The command is used to create new user accounts.

**Syntax:** useradd <options> user\_name

* **sudo useradd user\_name:** It creates a user with the specific name.
* **sudo useradd -m user\_name:** It creates user with the home directory.
* **sudo useradd -m -d /custom/home/user\_name user\_name:** It creates user with specify home directory.
* **Sudo useradd -u 1005 user\_name:** It creates a user with the specific user ID.
* **Sudo useradd** -g group\_name user\_name: It creates a user and assign the user to the specific group.
* **Sudo useradd** -G grp\_name1 grp\_name2 user\_name: Adds the user to the multiple groups.
* **Sudo usersadd** -e YYYY-MMM-D user\_name: It creates a user with the expire date.
* **Sudo useradd** -r systemuser: It creates a system user (used for the services or daemons).
* **Sudo useradd** -c “Kittu, a friend” user\_name: Adds the comment to the user.

1. **passwd**: The command is used to manage the user passwords. It allows the users to change their own passwords and administrators to change passwords of the other users or related settings.

**Syntax**: passwd <options> user\_name

* **passwd:** To change the password for the currently logged-in user.
* **sudo passwd user\_name:** To change the password for another user.
* **sudo passwd -l user\_name:** To lock a user account.
* **sudo passwd -u user\_name:** To unlock a previously locked user account.
* **sudo passwd -e user\_name:** To force a user to change their password the next time they log in.
* **sudo passwd -d user\_name:** To remove a user's password (making the account passwordless).
* **sudo passwd -x 90 user\_name:** To set the maximum number of days a password is valid.
* **sudo passwd -n 7 user\_name:** To set the minimum number of days required between password changes.
* **sudo passwd -i 10 user\_name:** To set the number of days after password expiration before the account is locked.
* **sudo passwd -S user\_name:** To view the password status for a user.

1. **usermod:** The usermod command in Linux is used to modify existing user accounts. It allows administrators to change various attributes of a user, such as their username, home directory, login shell, group ,membership and more.

* **sudo usermod -l newusername oldusername**: To rename an existing user.
* **sudo usermod -u newUID username:** To change the User ID (UID) of a user.
* **sudo usermod -aG group1,group2 username**: To add a user to additional (supplementary) groups.
* **sudo usermod -d /new/home/dir -m username:** To change the home directory of a user.
* **sudo usermod -L username:** To lock a user account (prevent login).
* **sudo usermod -U username:** To unlock a user account.
* **sudo usermod -e YYYY-MM-DD username:** To set an expiration date for a user account.
* **sudo usermod -c "John Doe, Developer" username:** To add or modify the comment field (e.g., full name or description).

1. **userdel:** The userdel command in Linux is used to delete user accounts from the system. It removes the user's entry from the /etc/passwd, /etc/shadow, /etc/group,and /etc/gshadow files.

**Syntax:** userdel [options] username

* **sudo userdel user\_name**: To delete a user account.
* **sudo userdel -r user\_name**: To delete a user account along with their home directory.
* **sudo userdel -f user\_name:** To force delete a user account, even if the user is logged in.
* **sudo userdel -r -f user\_name:** To delete a user account and remove all files owned by the user.

1. **chage:** The chage command in Linux is used to manage user password aging and expiration policies. It allows administrators to set or modify password expiration dates, warning periods, and other related settings. These settings are stored in the /etc/shadow file.

**Syntax:** chage <options> user\_name

* **sudo chage -l user\_name:** To view the current password aging information for a user.
* **sudo chage -E 2024-12-31 user\_name:** To set a specific date when the user's password will expire.
* **sudo chage -M 90 user\_name:** To set the maximum number of days a password is valid.
* **sudo chage -m 7 user\_name:** To set the minimum number of days required between password changes.
* **sudo chage -W 7 user\_name:** To set the number of days before password expiration that the user will receive a warning.
* **sudo chage -I 10 user\_name:** To set the number of days after password expiration before the account is locked.

1. **groups:** The groups command in Linux is used to display the groups to which a user belongs. It provides a simple way to check group memberships for the current user or a specified user.

**Syntax: groups user\_name**

* **groups:** To see the groups the currently logged-in user belongs to.
* **groups grp\_name:** To check the groups for another user.
* **groups usr\_name | grep grp\_name:** To verify if a user belongs to a specific group.

1. **groupadd:** The groupadd command in Linux is used to create new groups on the system. Groups are essential for managing file permissions and access control, as they allow multiple users to share common permissions.

**Syntax:** groupadd <options> grp\_name

* **sudo groupadd group\_name:** To create a new group.
* **sudo groupadd -g 1005 group\_name**: To create a group with a custom GID.
* **sudo groupadd -r systemgroup:** To create a system group (typically used for services or daemons).

1. **gpasswd:** It allows you to manage group passwords, add or remove users from groups, and assign group administrators**.**

**Syntax: gpasswd [options] groupname**

* **sudo gpasswd grp\_name**: To set a password for a group.
* **sudo gpasswd -r grp\_name:** To remove a group password.
* **sudo gpasswd -a usr\_name grp\_name**: To add a user to a group.
* **sudo gpasswd -d usr\_name grp\_name:** To remove a user from a group.
* **sudo gpasswd -A usr\_name grp\_name:** To assign a group administrator. Group administrators can add or remove users from the group and change the group password.
* **sudo gpasswd -M john,jane,alice grp\_name**: To add multiple users to a group.

1. **groupmod:** The command is used to modify existing groups on the system. It allows you to change a group's name, Group ID (GID), or other attributes.

**Syntax: groupmod <options> group\_name**

* **sudo groupmod -n newgroupname oldgroupname:** To rename an existing group.
* **sudo groupmod -g newGID groupname**: To change the Group ID (GID) of an existing group.
* **sudo groupmod -n newgroupname -g newGID oldgroupname:** To change both the name and GID of a group.

1. **groupdel:** The groupdel command in Linux is used to delete existing groups from the system. It removes the group entry from the /etc/group and /etc/gshadow files.

**Syntax: groupdel groupname**

* **sudo groupdel grp\_name**: To delete an existing group.
* **sudo groupdel systemgroup:** To delete a system group. System groups are typically used for services or daemons and have GIDs within the system range.

1. **id:** Use id to quickly check a user's UID, GID. For finding the use belongs to how many groups**.**

**Syntax:**

* **id user\_name:** Displays the groups of the user.

1. **lid:** Use lid to explore group membership details.

**Syntax:**

* **lid -g grp\_name**

**Remote Access Commands**

1. **ssh:** The ssh (Secure Shell) command in Linux is used to securely connect to remote systems over an encrypted network. It allows users to log in to remote machines, execute commands, and transfer files securely**.**

**Syntax:**

* **ssh user@hostname:** To connect to a remote server.
* **ssh -p port\_number user@hostname:** To connect to a remote server using a non-default port (default is 22).
* **ssh user@hostname command:** To run a command on a remote server without opening an interactive shell.
* **ssh -i /path/to/private\_key user@hostname:** To connect to a remote server using a private key.

1. **scp:** The scp (Secure Copy) command in Linux is used to securely copy files and directories between local and remote systems over an encrypted SSH connection. It is a simple and efficient way to transfer files securely.

**Syntax:**

* **scp /path/to/local/file user@remote\_host:/path/to/remote/directory:** To copy a file from your local machine to a remote server.
* **scp user@remote\_host:/path/to/remote/file /path/to/local/directory:** To copy a file from a remote server to your local machine.

**Hardware Information Command:**

1. **free:** The free command in Linux is used to display the amount of free and used memory (RAM and swap) in the system.

**Syntax:**

* **free -h:** To display memory usage in a human-readable format.
* **free -b:** To display memory usage in bytes.
* **free -k:** To display memory usage in kilobytes.
* **free -m:** To display memory usage in megabytes.
* **free -s interval:** To display memory usage continuously (refreshing every few seconds).
* **free -t:** To display only the total memory usage**.**

1. **dmidecode:** The dmidecode command in Linux is used to retrieve hardware information from the system's DMI (Desktop Management Interface) table, also known as SMBIOS (System Management BIOS). It provides detailed information about the system's hardware components, such as the motherboard, CPU, memory, BIOS, and more.

**Syntax:**

* **sudo dmidecode:** list of hardware details, including BIOS, system, motherboard, CPU, memory, and more.
* **sudo dmidecode -t type:** To display information for a specific DMI type.
* **sudo dmidecode –type:** To list all available DMI types and their descriptions.
* **sudo dmidecode -t system:** To display system information (e.g., manufacturer, product name, serial number).
* **sudo dmidecode -t bios:** To display BIOS information (e.g., vendor, version, release date).

1. **vmstat:** The vmstat (Virtual Memory Statistics) command in Linux is used to monitor system performance, including memory, processes, CPU activity, I/O, and more.

**Syntax:**

* **vmstat:** To display a single summary of system performance.
* **vmstat 2:** To display statistics continuously with a specified delay (e.g., every 2 seconds).

**Automating/Scheduling Tasks Commands**

1. **cron:** It is a daemon which is used to schedule tasks (commands or scripts) to run automatically at specified intervals. It is commonly used for automating repetitive tasks such as backups, system maintenance, log rotation, and more. It reads the /etc/crontab.

* crontab is a program which is used to install, uninstall or list the tables.

**Syntax:**

**# Minute(\*) Hour(\*) Day of Month(\*) Month Day of Week(\*) Command/script(\*)**

**# (0-59) (0-23) (1-31) (1-12) (0-6) /usr/bin/find**

* **# crontab -l:** Displays how many crontabs are configured.
* **# crontab -e:** The command is going to edit the crontab file in the default text editor.
* **# crontab -r:** Delete the crontab entries.
* **# 0 2 \* \* \* tar -czf /backup/backup.tar.gz /home/user/important\_files:** Schedule regular backups of important files or databases at every day 2 AM.
* **0 0 \* \* 0 rm -rf /tmp/\*:** Clean up temporary files or logs to free up disk space at Sunday mid night.
* **0 3 \* \* \* gzip /var/log/\*.log:** Rotate or compress log files to prevent them from growing too large at every day 3AM.
* **0 \* \* \* \* /usr/bin/python3 /path/to/script.py:** Execute custom scripts at regular intervals at every hour.
* **0 8 \* \* 1-5 echo "Don't forget the meeting!" | mail -s "Reminder"** [**user@example.com**](mailto:user@example.com): Send email notifications or alerts at specific times at every weekday at 8 AM.
* **0 18 \* \* \* /path/to/script.sh:** Run a script every day at 6 PM.
* **\*/5 \* \* \* \* /path/to/command:** Run a command every 5 minutes.
* **\* \* \* \* \* /path/to/command >> /path/to/logfile.log 2>&1: Redirect output to a log file for debugging.**
* **\* \* \* \* \* /path/to/command > /path/to/logfile.log : Redirect output to a log file.**

**Communication Commands:**

1. **mail:** The mail command in Linux is a simple utility used to send and receive emails from the command line.

**Syntax:**

* **mail -s "Subject"** [**recipient@example.com**](mailto:recipient@example.com)**:** After running the command, you can type the body of the email. Press Ctrl+D to send the email.
* **df -h | mail -s "Disk Usage Alert"** [**admin@example.com**](mailto:admin@example.com)**:** Send email notifications for system events like high CPU usage, disk space, or failed cron jobs.
* **mail -s "Log File" -a /var/log/syslog** [**admin@example.com**](mailto:admin@example.com)**:** Automatically send log files or reports via email.
* **echo "Meeting at 10 AM." | mail -s "Meeting Reminder"** [**user1@example.com,user2@example.com**](mailto:user1@example.com,user2@example.com)**:** Send the same email to multiple recipients.

**Other Commands:**

1. **clear:** The clear command in Linux is a simple yet essential utility used to clear the terminal screen.

**Syntax:**

* **clear:** This command will immediately clear the terminal screen.

2.**cal:** The cal command in Linux is a simple utility used to display a calendar in the terminal. It can show calendars for specific months, years, or a range of months and years.

**Syntax:**

* **cal:** Quickly check the current month's calendar.
* **cal 9 2023:** Display the calendar for a specific month and year.
* **cal 2024:** Display the entire calendar for a specific year.
* **cal -3:** Display multiple months in a single view.
* **cal -h:** Highlight the current date in the calendar.

3.**wget:** The wget command in Linux is a powerful, non-interactive tool used to download files from the web. It supports downloading via HTTP, HTTPS, and FTP protocols and is highly versatile, making it a favorite for automating downloads and mirroring websites.

**Syntax:**

* **wget** [**https://example.com/file.zip**](https://example.com/file.zip)**:** Download a file from a URL.
* **wget -O archive.zip** [**https://example.com/file.zip**](https://example.com/file.zip)**:** Save the downloaded file with a custom name archieve zip.
* **wget -c** [**https://example.com/file.zip**](https://example.com/file.zip)**:** Resume a partially downloaded file.

4. **tee:** The tee command in Linux is a versatile utility used to read from standard input (stdin) and write to both standard output (stdout) and one or more files simultaneously.

**Syntax:**

* **ls | tee output.txt:** Capture the output of a command and save it to a file while displaying it on the terminal.
* **ls | tee -a output.txt:** Append the output to a file instead of overwriting it.
* **ls | tee file1.txt file2.txt:** Write the output to multiple files simultaneously.

5. **script:** The script command in Linux is a powerful utility used to record or log everything that happens in a terminal session. It captures all input and output, including commands, their output, and even timing information.

**Syntax:**

* **script session.log:** Capture a terminal session for later review or sharing.
* **script tutorial.log:** Record terminal sessions to create tutorials or documentation.