//Akash Chandran

Introduction to Linux

who:

Displays a list of users currently logged into the system. Shows information such as the username, terminal, date and time of login, and the IP address or hostname of the remote host.

whoami:

Displays the current logged-in user's username. Useful for confirming the identity of the user executing commands, especially when using `sudo` or switching users.

pwd:

Prints the current working directory.Useful for determining the full path of the directory you are currently in.

date:

Displays the current date and time.Useful for checking the system's date and time settings, and can be formatted in various ways using options.

Example:

To display the current date and time in the default format:

```
date
```

To display the date in a custom format (e.g., YYYY-MM-DD):

```
date +"%Y-%m-%d"
```

ls:

Lists the contents of a directory. Useful for viewing files and directories within the current directory or a specified path.

Example:

To list all files and directories in the current directory:

```
ls
```

To list all files and directories, including hidden ones:

```
ls -a
```

To list files and directories with detailed information (permissions, owner, size, etc.):

```
ls -l
```

To list files and directories with detailed information, including hidden ones:

```
ls -la
```

To list files that match a specific pattern :

```
ls A[a-z0-9].txt
```

ps: Displays information about active processes.

Useful for monitoring and managing running processes on the system.

Example:

To display all running processes in a full-format listing:

ps -ef

To display processes related to a specific command (e.g., bash):

ps -ef | grep bash

grep:

Searches for patterns within files.Useful for finding specific text within files or output from other commands.

Example:

To search for a specific string (e.g., 'example') within a file (e.g., file.txt):

```
grep 'example' file.txt
```

To search for a pattern recursively within a directory:

```
grep -r 'pattern' /path/to/directory
```

To search for a pattern and display line numbers:

```
grep -n 'pattern' file.txt
```

To search for a pattern ignoring case sensitivity:

```
grep -i 'pattern' file.txt
```

To search for a pattern and display only the matching part of the line:

```
grep -o 'pattern' file.txt
```

cat:

Concatenates and displays the content of files.Useful for viewing the content of files, combining multiple files, and creating new files.

Example:

To display the content of a file (e.g., file.txt):

```
cat file.txt
```

To concatenate multiple files and display the output:

```
cat file1.txt file2.txt
```

To create a new file and write content to it:

```
cat > newfile.txt
```

(Type the content and press Ctrl+D to save)

To append content to an existing file:

```
cat >> existingfile.txt
```

(Type the content and press Ctrl+D to save)

ip a:

Displays information about all network interfaces.Useful for checking IP addresses, network interfaces, and their statuses.

Example:

To display detailed information about all network interfaces:

```
ip a
```

To display information about a specific interface (e.g., eth0):

```
ip a show eth0
```

cp:

Copies files and directories.Useful for duplicating files and directories to a new location.

Example:

To copy a file (e.g., file.txt) to a new location (e.g., /path/to/destination/):

```
cp file.txt /path/to/destination/
```

To copy a file and rename it:

```
cp file.txt /path/to/destination/newfile.txt
```

To copy a directory and its contents recursively:

```
cp -r /path/to/source_directory /path/to/destination_directory
```

To copy files interactively, prompting before overwrite:

```
cp -i file.txt /path/to/destination/
```

To copy files and preserve attributes (e.g., timestamps, ownership):

```
cp -p file.txt /path/to/destination/
```

To copy files verbosely, displaying the files being copied:

```
cp -v file.txt /path/to/destination/
```

To forcefully copy files, overwriting existing files without prompting:

```
cp -f file.txt /path/to/destination/
```

ls -ltr:

Lists the contents of a directory in long format, sorted by modification time in reverse order (oldest first).Useful for viewing detailed information about files and directories, with the most recently modified files at the bottom.

Example:

To list all files and directories in the current directory, sorted by modification time in reverse order:

```
ls -ltr
```

rm:

Removes files or directories.Useful for deleting unwanted files or directories.

Example:

To remove a file (e.g., file.txt):

```
rm file.txt
```

To remove multiple files:

```
rm file1.txt file2.txt
```

To remove a directory and its contents recursively:

```
rm -r /path/to/directory
```

To remove files interactively, prompting before each removal:

```
rm -i file.txt
```

To forcefully remove files, ignoring non-existent files and never prompting:

```
rm -f file.txt
```

To remove directories and their contents forcefully:

```
rm -rf /path/to/directory
```

To remove an empty directory (e.g., emptydir):

```
rm -d emptydir
```

find . -name:

Searches for files and directories by name within the current directory and its subdirectories.

Useful for locating files and directories based on their names.

Example:

To find a file named 'example.txt' within the current directory and its subdirectories:

```
find . -name 'example.txt'
```

find . -iname:

Searches for files and directories by name within the current directory and its subdirectories, ignoring case sensitivity.Useful for locating files and directories based on their names without considering case.

Example:

To find a file named 'example.txt' (case insensitive) within the current directory and its subdirectories:

```
find . -iname 'example.txt'
```

```
find . -iname Downloads -mtime -7:
```

Searches for files and directories named 'Downloads' within the current directory and its subdirectories that have been modified in the last 7 days.Useful for locating recently modified files and directories with a specific name.

Example:

To find a directory named 'Downloads' that has been modified in the last 7 days:

```
find . -iname 'Downloads' -mtime -7
```

```
find . -iname Downloads -mtime +7
```

Searches for files and directories named 'Downloads' within the current directory and its subdirectories that have been modified more than 7 days ago.Useful for locating files and directories with a specific name that have not been modified recently.

Example:

To find a directory named 'Downloads' that has been modified more than 7 days ago:

```
find . -iname 'Downloads' -mtime +7
```

```
find . -name "*.*" -exec grep "added" {} \;
```

Searches for files with any extension within the current directory and its subdirectories, and executes the `grep` command to search for the string "added" within those files.Useful for finding files containing a specific string.

Example:

To find files with any extension and search for the string "added" within those files:

```
find . -name "*.*" -exec grep "added" {} \;
```

sort:

Sorts the lines of a file or input.Useful for organizing data in a specific order.

Example:

To sort the lines of a file (e.g., a.txt) in ascending order:

```
sort a.txt
```

To sort the lines of a file in descending order:

```
sort -r a.txt
```

To sort the lines of a file numerically:

```
sort -n a.txt
```

To sort the lines of a file and remove duplicates:

```
sort -u a.txt
```

To sort the lines of a file based on a specific field (e.g., the second field):

```
sort -k 2 a.txt
```

df:

Displays disk space usage of file systems.Useful for checking available and used disk space on mounted file systems.

Example:

To display disk space usage in a human-readable format:

```
df -h
```

To display disk space usage of a specific file system (e.g., /dev/sda1):

```
df -h /dev/sda1
```

To display disk space usage including file system type:

```
df -T
```

To display disk space usage in inodes:

```
df -i
```

cut:

Removes sections from each line of files.Useful for extracting specific columns or fields from text files or command output.

Example:

To extract the first 10 characters of each line in a file (e.g., file.txt):

```
cut -c 1-10 file.txt
```

To extract the second and fourth fields from a file (e.g., file.txt) using a space as the delimiter:

```
cut -d ' ' -f 2,4 file.txt
```

To extract the first and third fields from a file (e.g., file.txt) using a comma as the delimiter:

```
cut -d ',' -f 1,3 file.txt
```

less:

Displays the content of files one screen at a time. Useful for viewing large files without loading the entire file into memory.

Example:

To view the content of a file (e.g., file.txt):

```
less file.txt
```

head:

Displays the first few lines of a file. Useful for quickly viewing the beginning of a file without opening the entire file.

Example:

To display the first 10 lines of a file (e.g., file.txt):

```
head file.txt
```

To display the first 20 lines of a file:

```
head -n 20 file.txt
```

tail:

Displays the last few lines of a file.Useful for quickly viewing the end of a file, especially log files.

Example:

To display the last 10 lines of a file (e.g., file.txt):

```
tail file.txt
```

To display the last 20 lines of a file:

```
tail -n 20 file.txt
```

top:

Displays real-time information about system processes. Useful for monitoring system performance, resource usage, and managing processes.

kill:

Sends a signal to a process, usually to terminate it. Useful for stopping processes that are running in the background or are unresponsive.

tar:

Archives and compresses files and directories. Useful for creating backups, transferring files, and reducing storage space.

Example:

To create an archive (e.g., archive.tar) from a directory (e.g., /path/to/directory):

```
tar -cvf archive.tar /path/to/directory
```

To extract an archive (e.g., archive.tar):

```
tar -xvf archive.tar
```

To create a compressed archive using gzip (e.g., archive.tar.gz):

```
tar -czvf archive.tar.gz /path/to/directory
```

To extract a compressed archive using gzip (e.g., archive.tar.gz):

```
tar -xzvf archive.tar.gz
```

du:

Displays disk usage of files and directories. Useful for checking the amount of disk space used by files and directories.

Example:

To display disk usage of the current directory:

```
du
```

To display disk usage in a human-readable format:

```
du -h
```

To display disk usage of all files and directories, including subdirectories:

```
du -a
```

To display only the total disk usage of a directory:

```
du -s
```

ifconfig:

Configures and displays network interface parameters. Useful for viewing and modifying network interface settings, such as IP addresses and netmasks.

netstat:

Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.Useful for monitoring and troubleshooting network connections and performance.

wget:

Downloads files from the web.Useful for retrieving files from HTTP, HTTPS, and FTP servers.

curl:

Transfers data from or to a server using various protocols (HTTP, HTTPS, FTP, etc.).Useful for downloading or uploading files, interacting with APIs, and testing endpoints.

alias:

Creates shortcuts for commands. Useful for simplifying and customizing command-line usage.

To remove an alias:

```
unalias myalias
```

To list all currently defined aliases:

```
alias
```

systemctl:

Manages systemd services.Useful for starting, stopping, restarting, enabling, and disabling services on a systemd-based system.

Example:

To start a service (e.g., httpd):

```
systemctl start httpd
```

To stop a service:

```
systemctl stop httpd
```

To restart a service:

```
systemctl restart httpd
```

To enable a service to start at boot:

```
systemctl enable httpd
```

To disable a service from starting at boot:

```
systemctl disable httpd
```

To check the status of a service:

```
systemctl status httpd
```

To list all services:

```
systemctl list-units --type=service
```

crontab:

Schedules and manages recurring tasks.Useful for automating repetitive tasks by running commands or scripts at specified times and intervals.

Example:

To edit the crontab file for the current user:

```
crontab -e
```

To view the current user's crontab file:

```
crontab -l
```

To remove the current user's crontab file:

```
crontab -r
```

tr:

Translates or deletes characters from the input. Useful for transforming text by replacing or removing specific characters.

Example:

To translate all lowercase letters to uppercase:

```
echo "hello world" | tr 'a-z' 'A-Z'
```

To delete all digits from the input:

```
echo "hello123" | tr -d '0-9'
```

To replace spaces with underscores:

```
echo "hello world" | tr ' ' '_'
```

To compress consecutive repeated characters into a single character:

```
echo "aaabbbccc" | tr -s 'a-z'
```

sed:

Stream editor for filtering and transforming text. Useful for performing basic text transformations on an input stream (a file or input from a pipeline).

scp:

Securely copies files between hosts over a network.Useful for transferring files to and from remote servers securely.

Example:

To copy a file (e.g., file.txt) from the local system to a remote system:

```
scp file.txt user@remote_host:/path/to/destination/
```

To copy a file from a remote system to the local system:

```
scp user@remote_host:/path/to/file.txt /path/to/destination/
```

To copy a directory and its contents recursively from the local system to a remote system:

```
scp -r /path/to/source_directory
user@remote_host:/path/to/destination_directory
```

To copy a file from a remote system to another remote system:

```
scp user1@remote_host1:/path/to/file.txt
user2@remote_host2:/path/to/destination/
```

awk:

A powerful text processing and pattern scanning language.Useful for extracting and manipulating data from text files or command output.

Example:

To print the second column of a file (e.g., file.txt):

```
awk '{print $2}' file.txt
```

To print lines where the third column is greater than 100:

```
awk '$3 > 100' file.txt
```

To print the sum of the values in the first column:

```
awk '{sum += $1} END {print sum}' file.txt
```

To print lines matching a specific pattern (e.g., 'error'):

```
awk '/error/' file.txt
```

To use a custom field delimiter (e.g., comma):

```
awk -F ',' '{print $1}' file.csv
```