Top 50 Linux Commands You MUST Know

Introduction, Options and Examples

Linux Support

DigiTalk

https://digitalksystems.com/

Reach us at digitalk.fmw@gmail.com
DigiTalk Channel: https://www.youtube.com/channel/UCCGTnI9vvF_ETMhGUXGdFWw
Playlists: https://www.youtube.com/@digitalk.middleware/playlists
Weblogic Server Architecture: https://youtu.be/gNqeIfLjUqw

DigiTalk Udemy Courses and

Coupon Code (Embedded in URL)

SOA Suite Administration

https://www.udemy.com/course/mastering-oracle-soa-suite-12c-administration/?couponCode=3748CA8CCCF4A124B4E9

JBoss 8 Administration

https://www.udemy.com/course/mastering-jboss-eap-8-administration-from-intro-to-advanced/?couponCode=C0947AF96757C942530F

OHS Administration

https://www.udemy.com/course/mastering-oracle-ohs-http-12c-web-server-administration/?couponCode=6203B4E94AA374CFA326

Weblogic Server Administration

https://www.udemy.com/course/oracle-weblogic-server-12c-and-14c-administration/?couponCode=D6E8B65B3FACB040D423

You can write us on digitalk.fmw@gmail.com if coupon code expired.

1. ls

Introduction: Lists directory contents.

Options:
-l: Long format.

-a: Show hidden files.

Examples:

1. ls - List files in the current directory.
2. ls -l - Detailed list including file permissions and sizes.
3. ls -a - Show all files, including hidden ones.
4. ls -lh - Long format with human-readable file sizes.
5. ls /path - List files in a specific path.
6. ls -R - List files recursively.

2. cd

Introduction: Changes the current directory.

Options: No options.

Examples:

1. cd /path/to/directory - Change to a specific directory.
2. cd .. - Move up one directory.
3. cd ~ - Change to the home directory.
4. cd - - Change to the previous directory.
5. cd / - Change to the root directory.
6. cd ../.. - Move up two directories.

3. cat

Introduction: Concatenates and displays file content.

Options:

-n: Number all output lines.

-b: Number non-empty lines.

Examples:

1. cat file.txt - Display content of file.txt.
2. cat -n file.txt - Display content with line numbers.
3. cat file1.txt file2.txt - Concatenate and display multiple files.
4. cat > file.txt - Write to file.txt.
5. cat file.txt | less - View content with paging.
6. cat file.txt | grep 'search' - Search for 'search' in file content.

4. pwd

Introduction: Prints the current working directory.

Options: No options.

Examples:

1. pwd - Display the full path of the current directory.
2. pwd -P - Display the physical directory (resolves symbolic links).
3. pwd -L - Display the logical directory.
4. echo $(pwd) - Use in scripts to display the current path.
5. cd /path/to/dir; pwd - Show path after changing directories.
6. pwd | tee path.txt - Save the current directory path to a file.

5. ps

Introduction: Displays information about running processes.

Options:

-e: Show all processes.

-f: Full format listing.

Examples:

1. ps - Display processes for the current shell.
2. ps -e - List all processes.
3. ps -ef - Detailed list of all processes.
4. ps aux - Detailed process information in BSD format.
5. ps -u username - List processes for a specific user.
6. ps -o pid,cmd - Display process ID and command.

Advance Examples

List All Processes with Detailed Information

```
ps aux
```

Explanation: Lists all processes running on the system with detailed information, including user, PID, CPU, and memory usage.

List Processes Tree View

```
ps aux --forest
```

Explanation: Shows processes in a tree-like format, depicting parent-child relationships between processes.

Filter Processes by User

```
ps -u username
```

Explanation: Lists processes running under a specific user username.

Display Process Information with CPU and Memory Usage

```
ps -eo pid,comm,%cpu,%mem --sort=-%cpu
```

Explanation: Lists all processes with PID, command name, CPU, and memory usage, sorted by CPU usage in descending order.

Show Process Tree for a Specific Process

```
pstree -p <pid>
```

Explanation: Displays the process tree for the process with PID <pid>, showing the hierarchy of processes.

List Processes by Their Command Line Arguments

```
ps -ef | grep 'pattern'
```

Explanation: Lists processes with details, then filters the output to include only lines containing pattern in their command line arguments.

Display Only the Process IDs of Running Processes

```
ps -e -o pid
```

Explanation: Lists only the process IDs of all running processes.

Show Processes with a Specific Status (e.g., Sleeping)

```
ps -e -o pid,stat | grep 'S'
```

Explanation: Lists processes with their statuses, then filters to show only those with a 'Sleeping' status (S).

Display Processes Running for a Specific Terminal

```
ps -t tty1
```

Explanation: Lists processes associated with terminal tty1.

Monitor Process Resource Usage in Real-Time

```
watch 'ps aux --sort=-%mem | head -n 10'
```

Explanation: Continuously monitors and displays the top 10 processes by memory usage in real-time using watch.

6. mkdir

Introduction: Creates directories.

Options:

-p: Create parent directories as needed.

Examples:

1. `mkdir newdir` - Create a directory named newdir.
2. `mkdir -p parent/child` - Create nested directories.
3. `mkdir -m 755 dir` - Create a directory with specific permissions.
4. `mkdir /path/to/dir` - Create a directory at a specific path.
5. `mkdir -v dir` - Verbosely show directory creation.

6. mkdir --parents /path/to/dir - Same as -p, create parent directories.

7. cp

Introduction: Copies files or directories.

Options:

-r: Recursive copy for directories.

-i: Prompt before overwriting.

Examples:

1. cp file1.txt file2.txt - Copy file1.txt to file2.txt.
2. cp -r dir1 dir2 - Copy directory dir1 to dir2.
3. cp -i file1.txt file2.txt - Prompt before overwriting.
4. cp -u file1.txt dir/ - Copy only if file1.txt is newer.
5. cp -v file1.txt file2.txt - Verbosely show the copy process.
6. cp -a dir1 dir2 - Archive mode (preserves attributes).

8. mv

Introduction: Moves or renames files or directories.

Options:

-i: Prompt before overwriting.

-u: Move only if the source is newer.

Examples:

1. mv file1.txt file2.txt - Rename file1.txt to file2.txt.
2. mv file.txt /path/to/dir/ - Move file.txt to a directory.
3. mv -i file1.txt /path/to/dir/ - Prompt before overwriting.
4. mv -u file1.txt /path/to/dir/ - Move only if file1.txt is newer.
5. mv dir1/ dir2/ - Move dir1 to dir2.
6. mv -v file1.txt /path/to/dir/ - Verbosely show the move process.

9. rm

Introduction: Removes files or directories.

Options:

-r: Recursive removal for directories.

-f: Force removal without prompting.

Examples:

1. rm file.txt - Remove file.txt.
2. rm -r dir/ - Remove directory dir and its contents.
3. rm -f file.txt - Force remove file.txt.

4. rm -rf dir/ - Force remove directory dir and its contents.
5. rm -i file.txt - Prompt before removing.
6. rm -v file.txt - Verbosely show the removal process.

10. wc

Introduction: Counts lines, words, and characters in a file.

Options:

-l: Count lines.

-w: Count words.

-c: Count characters.

Examples:

1. wc file.txt - Count lines, words, and characters in file.txt.
2. wc -l file.txt - Count lines in file.txt.
3. wc -w file.txt - Count words in file.txt.
4. wc -c file.txt - Count characters in file.txt.
5. echo "test" | wc -w - Count words from the input.
6. wc -l *.txt - Count lines in all .txt files.

11. whoami

Introduction: Displays the currently logged-in user.

Options: No options.

Examples:

1. whoami - Show the current user.
2. sudo whoami - Show the user after switching to superuser.
3. whoami | tee user.txt - Save the current user to a file.
4. echo "Current user: $(whoami)" - Include the user in a message.
5. whoami | grep 'user' - Search for the user name.
6. sudo -u anotheruser whoami - Show the user name when running as another user.

12. head

Introduction: Displays the beginning of a file.

Options:

-n: Specify the number of lines.

Examples:

1. head file.txt - Display the first 10 lines of file.txt.
2. head -n 20 file.txt - Display the first 20 lines.
3. head -c 100 file.txt - Display the first 100 bytes.

4. head -n 10 *.log - Display the first 10 lines of all .log files.
5. head -n 5 file.txt | grep 'pattern' - Search in the first 5 lines.
6. head -n 10 file.txt | less - View the first 10 lines with paging.

13. tail

Introduction: Displays the end of a file.

Options:

-n: Specify the number of lines.

-f: Follow the file as it grows.

Examples:

1. tail file.txt - Display the last 10 lines of file.txt.
2. tail -n 20 file.txt - Display the last 20 lines.
3. tail -f file.txt - Continuously display new lines added to file.txt.
4. tail -n 50 /var/log/syslog - Display the last 50 lines of syslog.
5. tail -f /var/log/messages | grep 'error' - Follow and filter error lines in real-time.
6. tail -c 100 file.txt - Display the last 100 bytes.

14. ln

Introduction: Creates hard and symbolic links.

Options:

-s: Create a symbolic link.

Examples:

1. ln file1.txt link.txt - Create a hard link.
2. ln -s file1.txt symlink.txt - Create a symbolic link.
3. ln -s /path/to/file symlink - Create a symbolic link to a file in a different directory.
4. ln -s /path/to/dir symlink - Create a symbolic link to a directory.
5. ln -sf file1.txt symlink - Force create a symbolic link, overwriting if necessary.
6. ln -s ../file1.txt symlink - Create a relative symbolic link.

15. kill

Introduction: Sends signals to processes.

Options:

-9: Force kill the process.

-l: List signal names.

Examples:

1. kill PID - Send the default signal (TERM) to process with ID PID.
2. kill -9 PID - Force kill the process with ID PID.
3. kill -l - List all available signal names.
4. kill -s HUP PID - Send the HUP signal to process PID.

5. kill -TERM PID - Send the TERM signal (default) to process PID.
6. kill -KILL PID - Send the KILL signal (force kill) to process PID.

16. sudo

Introduction: Executes commands as another user, usually superuser.

Options:

-u: Specify the user to run the command as.

-i: Start a login shell.

Examples:

1. sudo ls - List files as superuser.
2. sudo -u username command - Run a command as a specific user.
3. sudo -i - Start a login shell as root.
4. sudo cp file.txt /root/ - Copy file to root's directory.
5. sudo -l - List commands you can run with sudo.
6. sudo visudo - Edit the sudoers file.

17. alias

Introduction: Creates shortcuts for commands.

Options: No options.

Examples:

1. alias ll='ls -la' - Create a shortcut for ls -la.
2. alias gs='git status' - Create an alias for git status.
3. alias rm='rm -i' - Create an alias to prompt before removing files.
4. alias cls='clear' - Create a shortcut to clear the terminal.
5. alias ..='cd ..' - Create an alias to move up one directory.
6. alias l='ls -CF' - Create a shortcut to list files in columns.

18. date

Introduction: Displays or sets the system date and time.

Options:

-u: Display or set UTC time.

+format: Display time in a specific format.

Examples:

1. date - Display the current date and time.
2. date -u - Display the current UTC time.
3. date '+%Y-%m-%d %H:%M:%S' - Display time in a specific format.
4. date -s '2024-08-07 12:00:00' - Set the system date and time.

5. date -d 'next Monday' - Display the date for next Monday.
6. date -R - Display the date in RFC 2822 format.

19. ssh

Introduction: Securely connects to remote machines.

Options:

-p: Specify port.

-i: Specify identity file.

Examples:

1. ssh user@hostname - Connect to a remote machine.
2. ssh -p 2222 user@hostname - Connect using a specific port.
3. ssh -i ~/.ssh/id_rsa user@hostname - Connect using a specific key.
4. ssh -X user@hostname - Enable X11 forwarding.
5. ssh -L local_port:remote_host:remote_port user@hostname - Create a port forward.
6. ssh -t user@hostname 'command' - Execute a command on the remote host.

Advance Examples:

Run a Command on a Remote Server and Get Output Locally

```
ssh user@remote-server 'ls -l /path/to/directory' > local_file.txt
```

Explanation: Executes ls -l /path/to/directory on remote-server and saves the output to local_file.txt on the local machine.

Copy Files to Remote Server Using SSH and scp

```
scp local_file.txt user@remote-server:/path/to/destination/
```

Explanation: Copies local_file.txt to /path/to/destination/ on remote-server.

Execute Commands on Multiple Remote Servers

```
for server in server1 server2 server3; do
  ssh user@$server 'uptime'
done
```

Explanation: Executes uptime on server1, server2, and server3.

Forward a Local Port to a Remote Server Port

```
ssh -L 8080:localhost:80 user@remote-server
```

Explanation: Forwards local port 8080 to port 80 on remote-server, allowing access to remote web services locally.

Run a Command on a Remote Server and Use grep to Filter the Output

```
ssh user@remote-server 'dmesg | grep error'
```

Explanation: Executes dmesg | grep error on remote-server to filter out error messages.

Perform Remote File Search with find and grep

```
ssh user@remote-server 'find /path/to/search -type f -name "*.log" -exec grep "search-term" {} +'
```

Explanation: Searches for search-term in .log files located in /path/to/search on the remote server.

Copy a Directory Recursively to a Remote Server

```
scp -r local_directory user@remote-server:/path/to/destination/
```

Explanation: Recursively copies local_directory to /path/to/destination/ on remote-server.

Execute a Command on a Remote Server and Pipe the Output to a Local File

```
ssh user@remote-server 'cat /path/to/remote/file' | tee local_copy.txt
```

Explanation: Executes cat /path/to/remote/file on remote-server and saves the output to local_copy.txt.

Establish an SSH Tunnel for Secure Data Transfer

```
ssh -R 9000:localhost:3306 user@remote-server
```

Explanation: Creates an SSH tunnel from port 9000 on remote-server to port 3306 on the local machine (often used for database access).

Execute a Remote Command and Automatically Use ssh-keygen to Avoid Password Prompts

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -N "" && ssh-copy-id user@remote-server
```

Explanation: Generates a new SSH key pair and copies the public key to remote-server to allow password-less login.

20. diff

Introduction: Compares files line by line.

Options:

-u: Unified format.

-r: Recursively compare directories.

Examples:

1. `diff file1.txt file2.txt` - Compare two files.
2. `diff -u file1.txt file2.txt` - Unified format comparison.
3. `diff -r dir1 dir2` - Recursively compare directories.
4. `diff -q file1.txt file2.txt` - Report if files differ.
5. `diff -y file1.txt file2.txt` - Side-by-side comparison.
6. `diff -c file1.txt file2.txt` - Context format comparison.

21. cmp

Introduction: Compares two files byte by byte.

Options:

-l: Print the differing bytes.

-s: Suppress output, return only exit status.

Examples:

1. cmp file1.txt file2.txt - Compare two files.
2. cmp -l file1.txt file2.txt - Print differing bytes.
3. cmp -s file1.txt file2.txt - Compare silently (return status only).
4. cmp -i 100 file1.txt file2.txt - Start comparison at byte 100.
5. cmp -b file1.txt file2.txt - Print the byte differences.
6. cmp -n 50 file1.txt file2.txt - Compare the first 50 bytes.

22. uname

Introduction: Displays system information.

Options:

-a: All information.

-r: Kernel version.

Examples:

1. uname - Display the system name.
2. uname -a - Display all system information.
3. uname -r - Display the kernel version.
4. uname -m - Display machine hardware name.
5. uname -s - Display the kernel name.
6. uname -v - Display the kernel version.

23. clear

Introduction: Clears the terminal screen.

Options: No options.

Examples:

1. clear - Clear the terminal screen.
2. echo -e "\033c" - Alternative way to clear the terminal.
3. reset - Reset the terminal, similar to clear.
4. clear; ls - Clear the screen and then list files.
5. clear; echo "Cleared" - Clear and display a message.
6. ctrl+L - Shortcut to clear the terminal screen.

24. zip

Introduction: Compresses files into a ZIP archive.

Options:

-r: Recursive compression.

-e: Encrypt the archive.

Examples:

1. zip archive.zip file1.txt - Create a ZIP archive of file1.txt.
2. zip -r archive.zip dir/ - Recursively compress directory dir.
3. zip -e archive.zip file1.txt - Create an encrypted ZIP archive.
4. zip -q archive.zip file1.txt - Quiet mode, suppress output.
5. zip -u archive.zip file2.txt - Update the archive with file2.txt.
6. zip -d archive.zip file1.txt - Remove file1.txt from the archive.

25. unzip

Introduction: Extracts files from a ZIP archive.

Options:

-d: Specify destination directory.

-l: List contents without extracting.

Examples:

1. unzip archive.zip - Extract all files from archive.zip.
2. unzip -d /path/to/dir archive.zip - Extract files to a specific directory.
3. unzip -l archive.zip - List contents of the ZIP archive.
4. unzip -o archive.zip - Overwrite files without prompting.
5. unzip -q archive.zip - Quiet mode, suppress output.
6. unzip -x file.txt archive.zip - Extract all except file.txt.

26. gzip

Introduction: Compresses files using the GNU zip algorithm.

Options:

-d: Decompress.

-c: Output to stdout.

Examples:

1. gzip file.txt - Compress file.txt into file.txt.gz.
2. gzip -d file.txt.gz - Decompress file.txt.gz to file.txt.
3. gzip -c file.txt > file.txt.gz - Compress and output to file.txt.gz.
4. gzip -v file.txt - Verbosely display compression details.
5. gzip -l file.txt.gz - List information about file.txt.gz.

6. gzip -k file.txt - Keep the original file after compression.

27. gunzip

Introduction: Decompresses files compressed with gzip.

Options:

-c: Output to stdout.

-l: List contents of the archive.

Examples:

1. gunzip file.txt.gz - Decompress file.txt.gz to file.txt.
2. gunzip -c file.txt.gz > file.txt - Decompress and output to file.txt.
3. gunzip -d file.txt.gz - Alternative way to decompress.
4. gunzip -l file.txt.gz - List contents of file.txt.gz.
5. gunzip -v file.txt.gz - Verbosely display decompression details.
6. gunzip -k file.txt.gz - Keep the original file after decompression.

Advance Examples:

Decompress Files and Move Them to a Different Directory

```
gunzip -c file.gz | mv -t /path/to/directory
```

Explanation: Decompresses file.gz and moves the decompressed file to /path/to/directory.

Decompress Multiple Files and List the Files in the Directory

```
gunzip *.gz && ls -l
```

Explanation: Decompresses all .gz files in the current directory and then lists the files.

Decompress Files and Check for Errors with gzip -t

```
gunzip -c file.gz | gzip -t
```

Explanation: Decompresses file.gz and checks the integrity of the output.

Decompress a File and View Its Content with less

```
gunzip -c file.gz | less
```

Explanation: Decompresses file.gz and pipes the output to less for viewing.

Decompress Files and Search for a Pattern

```
gunzip -c file.gz | grep 'search-term'
```

Explanation: Decompresses file.gz and searches for search-term in the output.

Decompress Files and Count the Number of Lines

```
gunzip -c file.gz | wc -l
```

Explanation: Decompresses file.gz and counts the number of lines in the decompressed output.

Decompress Files and Replace a String with sed

```
gunzip -c file.gz | sed 's/old-string/new-string/g' > new_file
```

Explanation: Decompresses file.gz, replaces old-string with new-string using sed, and saves the result to new_file.

Decompress and Find Specific Files in the Directory

```
gunzip *.gz && find /path/to/directory -type f -name '*.txt'
```

Explanation: Decompresses all .gz files and then finds all .txt files in /path/to/directory.

Decompress Files and Sort the Output

```
gunzip -c file.gz | sort > sorted_file
```

Explanation: Decompresses file.gz, sorts the output, and saves it to sorted_file.

28. chmod

Introduction: Changes file permissions.

Options:

-R: Recursive change.

+x: Add execute permission.

Examples:

1. `chmod 755 file.txt` - Set permissions to rwxr-xr-x.
2. `chmod +x script.sh` - Add execute permission to script.sh.
3. `chmod -R 755 dir/` - Recursively set permissions for a directory.
4. `chmod u+w file.txt` - Add write permission for the user.
5. `chmod o-r file.txt` - Remove read permission for others.
6. `chmod a=r file.txt` - Set read-only permission for all users.

29. chown

Introduction: Changes file ownership.

Options:

-R: Recursive change.

:group: Change group ownership.

Examples:

1. `chown user file.txt` - Change owner of file.txt to user.
2. `chown user:group file.txt` - Change owner and group of file.txt.
3. `chown -R user:group dir/` - Recursively change ownership for a directory.
4. `chown :group file.txt` - Change group ownership only.
5. `chown user: file.txt` - Change only the owner.

6. `chown user:group file1.txt file2.txt` - Change ownership of multiple files.

30. grep

Introduction: Searches for patterns in files.

Options:

-i: Case-insensitive search.

-r: Recursive search.

Examples:

1. `grep 'pattern' file.txt` - Search for 'pattern' in file.txt.
2. `grep -i 'pattern' file.txt` - Case-insensitive search.
3. `grep -r 'pattern' dir/` - Recursively search for 'pattern' in a directory.
4. `grep -v 'pattern' file.txt` - Invert match (show lines not containing 'pattern').
5. `grep -n 'pattern' file.txt` - Show line numbers with matches.
6. `grep -E 'pattern1|pattern2' file.txt` - Extended regex search.

Advance Examples:

Search for a Pattern in Files Modified in the Last 24 Hours

```
find /path/to/directory -mtime -1 -type f -exec grep 'search-term' {} +
```

Explanation: Finds files modified in the last 24 hours and searches for search-term within them.

Search for a Pattern and Display the File Name Only

```
grep -l 'search-term' *.log
```

Explanation: Lists filenames that contain search-term in .log files.

Search for a Pattern and Display Line Numbers and Context

```
grep -n -C 5 'search-term' file.txt
```

Explanation: Shows line numbers and 5 lines of context around each match of search-term.

Search for a Pattern in Files with a Specific Extension and Exclude Certain Files

```
grep --exclude='*.log' -r 'search-term' /path/to/directory
```

Explanation: Recursively searches for search-term in files within the directory, excluding .log files.

Search for a Pattern and Display Only the Matching Part of the Line

```
grep -o 'pattern[0-9]+' file.txt
```

Explanation: Displays only the parts of the line that match the pattern.

Search for a Pattern and Replace the String (Use with sed for Replacement)

```
grep -rl 'old-string' /path/to/files | xargs sed -i 's/old-string/new-string/g'
```

Explanation: Finds files containing old-string and replaces it with new-string using sed.

Count Matching Lines Across Multiple Files

```
grep -c 'search-term' *.txt | awk -F: '{sum += $2} END {print sum}'
```

Explanation: Counts the total number of lines containing search-term across multiple .txt files.

Search for a Pattern in Files Larger Than 1GB

```
find /path/to/directory -size +1G -type f -exec grep 'search-term' {} +
```

Explanation: Finds files larger than 1GB and searches for search-term within them.

Search for a Pattern and Sort the Results

```
grep 'search-term' *.log | sort
```

Explanation: Searches for search-term in .log files and sorts the results.

Search for a Pattern in Multiple File Types

```
grep 'search-term' *.txt *.log
```

Explanation: Searches for search-term in both .txt and .log files.

Find Files Matching a Pattern and Then Search Inside

```
find /path/to/directory -name '*.txt' -exec grep 'search-term' {} +
```

Explanation: Finds .txt files and searches for search-term inside them.

Search for a Pattern and Show Only the Lines Matching Specific Fields (Using awk for Field Extraction)

```
grep 'search-term' file.txt | awk '{print $1, $2}'
```

Explanation: Searches for search-term and shows only the first and second fields of each matching line.

Show Lines Matching a Pattern and their Relative Line Numbers

```
grep -n 'search-term' file.txt | awk -F: '{print "Line " $1 ": " $2}'
```

Explanation: Shows lines matching search-term along with their line numbers, formatted with "Line".

Search for a Pattern and Save Results to a File

```
grep 'search-term' *.log > results.txt
```

Explanation: Searches for search-term in .log files and saves the results to results.txt.

31. tar

Introduction: Archives files and directories.

Options:

-c: Create a new archive.

-x: Extract from an archive.

-f: Specify archive file name.

Examples:

1. `tar -cvf archive.tar file1.txt file2.txt` - Create a tar archive.
2. `tar -xvf archive.tar` - Extract files from a tar archive.
3. `tar -tf archive.tar` - List contents of a tar archive.
4. `tar -czvf archive.tar.gz dir/` - Create a gzipped tar archive.
5. `tar -xzf archive.tar.gz` - Extract a gzipped tar archive.
6. `tar -C /path -xvf archive.tar` - Extract to a specific directory.

Advance Examples:

Create a Compressed Archive with gzip

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

Explanation: Creates a compressed archive using gzip with the name archive.tar.gz for the specified directory.

Create a Compressed Archive with bzip2

```
tar -cjvf archive.tar.bz2 /path/to/directory
```

Explanation: Creates a compressed archive using bzip2 with the name archive.tar.bz2.

Create a Compressed Archive with xz

```
tar -cJvf archive.tar.xz /path/to/directory
```

Explanation: Creates a compressed archive using xz with the name archive.tar.xz.

Extract a Specific File from a tar Archive

```
tar -xvf archive.tar.gz specific-file.txt
```

Explanation: Extracts only specific-file.txt from the archive.tar.gz archive.

Extract All Files Except a Specific File

```
tar -xvf archive.tar.gz --exclude='specific-file.txt'
```

Explanation: Extracts all files from the archive except specific-file.txt.

List Contents of a tar Archive

```
tar -tvf archive.tar.gz
```

Explanation: Lists the contents of the archive.tar.gz archive without extracting them.

Extract Files to a Specific Directory

```
tar -xvf archive.tar.gz -C /path/to/destination/
```

Explanation: Extracts files from archive.tar.gz into the specified directory /path/to/destination/.

Create an Archive of Multiple Directories

```
tar -czvf archive.tar.gz /dir1 /dir2 /dir3
```

Explanation: Creates a compressed archive with multiple directories (/dir1, /dir2, and /dir3).

Append Files to an Existing Archive

```
tar -rvf archive.tar file-to-add.txt
```

Explanation: Appends file-to-add.txt to the existing archive.tar.

Create a Tar Archive of Files Modified in the Last 24 Hours

```
find /path/to/directory -mtime -1 -print0 | tar -czvf archive.tar.gz --null -T -
```

Explanation: Finds and archives files modified in the last 24 hours into archive.tar.gz.

Exclude Multiple Files from an Archive

```
tar -czvf archive.tar.gz /path/to/directory --exclude='*.log' --exclude='*.tmp'
```

Explanation: Creates an archive excluding all .log and .tmp files.

Create a Split Archive (e.g., 100MB per Part)

```
tar -czvf - /path/to/directory | split -b 100M - archive.tar.gz.part
```

Explanation: Creates a compressed archive and splits it into parts of 100MB each.

Extract an Archive with Verbose Output

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

Explanation: Extracts a gzip-compressed archive with detailed output of the files being extracted.

Combine and Compress Using a Single Command

```
tar -cvf - /path/to/directory | gzip > archive.tar.gz
```

Explanation: Combines and compresses the directory in a single pipeline command.

List Files in an Archive with Specific Extensions

```
tar -tvf archive.tar.gz | grep '\.txt$'
```

Explanation: Lists only .txt files within the archive.tar.gz archive.

32. df

Introduction: Displays disk space usage.

Options:

-h: Human-readable format.

-T: Show filesystem type.

Examples:

1. df - Display disk space usage.
2. df -h - Human-readable format (e.g., GB, MB).
3. df -T - Show filesystem type.
4. df -i - Show inode usage.
5. df -H - Human-readable format with powers of 1000.
6. df /path - Show disk usage for a specific path.

Advance Examples:

Show Disk Usage with Human-Readable Sizes

```
df -h
```

Explanation: Displays disk space usage in a human-readable format (e.g., GB, MB).

Show Disk Usage for a Specific File System

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

Explanation: Displays disk usage statistics for the specified file system, such as /dev/sda1.

Display Disk Space Usage for All Mounted Filesystems

```
df -a
```

Explanation: Shows disk space usage for all mounted filesystems, including pseudo, duplicate, and inaccessible filesystems.

Show Disk Usage in Inodes

```
df -i
```

Explanation: Displays disk space usage in terms of inodes, which are data structures used to store information about files.

Show Disk Usage for a Specific Mount Point

```
df -h /home
```

Explanation: Provides disk usage statistics specifically for the /home mount point.

Display Disk Usage for All File Systems and Sort by Size

```
df -h | sort -k 2 -r
```

Explanation: Lists all file systems sorted by size in descending order.

Show Disk Usage Including Filesystems with 0% Used Space

```
df -h | awk '$5 == "0%"'
```

Explanation: Filters and displays filesystems that have 0% used space.

Display Disk Usage for a Specific File System and Exclude Filesystems with 100% Usage

```
df -h | awk '$5 != "100%"'
```

Explanation: Shows disk usage for filesystems excluding those that are 100% used.

Show Disk Usage and Exclude Certain File Systems (e.g., tmpfs)

```
df -h | grep -v '^tmpfs'
```

Explanation: Excludes tmpfs file systems from the disk usage output.

Display Disk Usage for All Filesystems with 1% or More Usage

```
df -h | awk '$5 ~ /[1-9][0-9]%/'
```

Explanation: Filters and shows filesystems with 1% or more usage.

Show Disk Usage for All Filesystems with More Than 90% Used Space

```
df -h | awk '$5 ~ /[9][0-9]%/'
```

Explanation: Displays filesystems where usage is greater than or equal to 90%.

Display Disk Usage with File System Type

```
df -Th
```

Explanation: Shows disk usage along with the file system type for each mounted file system.

Check Disk Space Usage on All Mounted File Systems with Usage Above 50%

```
df -h | awk '$5 ~ /[5-9][0-9]%/'
```

Explanation: Filters and displays filesystems with usage between 50% and 99%.

Show Disk Usage for a Specific Directory's File System

```
df -h $(dirname /path/to/directory)
```

Explanation: Displays disk usage for the file system where the specific directory resides.

Display Disk Usage and Only Show Filesystems That Are Mounted

```
df -h | grep -v 'tmpfs' | grep -v 'none'
```

Explanation: Excludes non-mounted filesystems and temporary filesystems from the output.

33. du

Introduction: Displays disk usage of files and directories.

Options:

-h: Human-readable format.

-s: Summarize total usage.

Examples:

1. du - Display disk usage for the current directory.
2. du -h - Human-readable format (e.g., GB, MB).
3. du -sh dir/ - Summarize total usage for dir.
4. du -a - Show usage for all files and directories.
5. du -m - Show usage in megabytes.
6. du -x - Avoid crossing filesystem boundaries.

Advance Examples:

Show Disk Usage of a Directory and Its Subdirectories

```
du -ah /path/to/directory
```

Explanation: Displays disk usage for the directory and its subdirectories, including hidden files, with human-readable sizes.

Show Only Directories Larger Than 1GB

```
du -h --max-depth=1 /path/to/directory | awk '$1 ~ /[0-9\.]+G/ {print $0}'
```

Explanation: Lists directories in /path/to/directory with sizes greater than or equal to 1GB.

Sort Directories by Size

```
du -ah /path/to/directory | sort -rh | less
```

Explanation: Lists all files and directories sorted by size in descending order, with human-readable sizes.

Find Largest Files in a Directory

```
du -ah /path/to/directory | grep -v '/$' | sort -rh | head -n 10
```

Explanation: Lists the 10 largest files (excluding directories) within /path/to/directory.

Show Disk Usage of Files Only (Excludes Directories)

```
du -ah /path/to/directory | grep -v '/$'
```

Explanation: Displays disk usage of files only, excluding directory entries.

Calculate Disk Usage of Files Modified in the Last 7 Days

```
find /path/to/directory -type f -mtime -7 -exec du -ch {} + | grep total$
```

Explanation: Finds and calculates the total disk usage of files modified in the last 7 days.

Show Disk Usage for All Directories with Depth of 2

```
du -h --max-depth=2 /path/to/directory
```

Explanation: Displays disk usage for directories up to a depth of 2 within /path/to/directory.

Find and Display Directories Exceeding a Specific Size

```
du -h --max-depth=1 /path/to/directory | awk '$1 ~ /[0-9\.]+G/ {print $0}'
```

Explanation: Lists directories within /path/to/directory that are 1GB or larger.

Show Disk Usage of Files in a Specific Directory and Sort by Size

```
du -ah /path/to/directory | sort -rh | awk '$1 ~ /[0-9\.]+M/ {print $0}'
```

Explanation: Lists and sorts files in /path/to/directory that are 1MB or larger.

Show Disk Usage for Files and Directories, Excluding Certain Patterns

```
du -ah /path/to/directory | grep -vE '(pattern1|pattern2)'
```

Explanation: Displays disk usage while excluding entries matching specific patterns (pattern1 or pattern2).

Summarize Disk Usage for All Subdirectories at a Depth of 3

```
du -h --max-depth=3 /path/to/directory
```

Explanation: Provides a summary of disk usage for subdirectories up to a depth of 3.

Show Disk Usage for Files Larger Than 100MB

```
du -ah /path/to/directory | awk '$1 ~ /[0-9\.]+M/ && $1 > 100 {print $0}'
```

Explanation: Lists files larger than 100MB within the specified directory.

Find Disk Usage of Hidden Files

```
du -ah /path/to/directory | grep '^\.[^/]*'
```

Explanation: Displays disk usage of hidden files (those starting with a dot) in the directory.

Display Total Disk Usage of a Directory and Subdirectories

```
du -sh /path/to/directory
```

Explanation: Shows the total disk usage of /path/to/directory including all its subdirectories.

Display Disk Usage for All Files with a Specific Extension

```
find /path/to/directory -type f -name '*.log' -exec du -ch {} + | grep total$
```

Explanation: Finds and calculates the total disk usage of all .log files within the directory.

34. top

Introduction: Displays real-time system processes and resource usage.

Options:

-d: Delay between updates.

-u: Show processes for a specific user.

Examples:

1. top - Display real-time process information.
2. top -d 5 - Update every 5 seconds.
3. top -u username - Show processes for a specific user.
4. top -p PID - Monitor a specific process ID.
5. top -n 1 - Show one iteration of process information.
6. top -b - Batch mode (suitable for logging).

Advance Examples:

Show Processes with CPU Utilization Above 50%

```
top -b -n 1 | awk '$9 > 50'
```

Explanation: Runs top in batch mode (-b) to capture a snapshot, then uses awk to filter processes with CPU usage above 50%. The %CPU field is usually the 9th column.

Show Processes with Memory Utilization Above 50%

```
top -b -n 1 | awk '$10 > 50'
```

Explanation: Captures a snapshot of top output and uses awk to filter processes with memory usage above 50%. The %MEM field is usually the 10th column.

Show Processes with CPU Utilization Between 30% and 70%

```
top -b -n 1 | awk '$9 >= 30 && $9 <= 70'
```

Explanation: Filters processes with CPU usage between 30% and 70%.

Show Processes with Memory Utilization Between 20% and 60%

```
top -b -n 1 | awk '$10 >= 20 && $10 <= 60'
```

Explanation: Filters processes with memory usage between 20% and 60%.

Show Top 5 Processes by CPU Utilization

```
top -b -n 1 | head -n 20 | grep -E '^ *[0-9]+' | sort -k 9 -r | head -n 5
```

Explanation: Extracts the top 5 processes by CPU usage by sorting the output from top based on the CPU column.

Show Top 5 Processes by Memory Utilization

```
top -b -n 1 | head -n 20 | grep -E '^ *[0-9]+' | sort -k 10 -r | head -n 5
```

Explanation: Extracts the top 5 processes by memory usage by sorting the output from top based on the memory column.

Filter Processes with Specific Command Name and High CPU Usage

```
top -b -n 1 | grep 'command_name' | awk '$9 > 50'
```

Explanation: Filters processes with the name command_name and CPU usage above 50%.

Display Processes with High I/O Wait Time

```
top -b -n 1 | awk '$12 > 10'
```

Explanation: Filters processes with I/O wait time (%wa) greater than 10%. The %wa field is usually the 12th column.

Show Processes with High System Time

```
top -b -n 1 | awk '$13 > 10'
```

Explanation: Filters processes with high system time (%st) greater than 10%. The %st field is usually the 13th column.

Monitor Specific Process ID (PID) and Filter by CPU Usage

```
top -b -n 1 | grep PID | awk '$9 > 50'
```

Explanation: Monitors the process with a specific PID and filters if its CPU usage is above 50%.

35. scp

Introduction: Securely copies files between hosts.

Options:

-r: Recursive copy.

-P: Specify port.

Examples:

1. `scp file.txt user@remote:/path` - Copy file to remote host.
2. `scp -r dir/ user@remote:/path` - Recursively copy directory.
3. `scp -P 2222 file.txt user@remote:/path` - Copy with a specific port.
4. `scp user@remote:/path/file.txt /local/path` - Copy from remote to local.
5. `scp -i ~/.ssh/id_rsa file.txt user@remote:/path` - Copy using a specific key.
6. `scp -v file.txt user@remote:/path` - Verbose mode.

Advance Examples:

Copy a Local File to a Remote Host

```
scp /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Copies localfile from the local machine to /path/to/remotefile on the remotehost.

Copy a Remote File to the Local Host

```
scp user@remotehost:/path/to/remotefile /path/to/localfile
```

Explanation: Copies remotefile from the remotehost to /path/to/localfile on the local machine.

Copy a Directory Recursively

```
scp -r /path/to/localdir user@remotehost:/path/to/remotedir
```

Explanation: Recursively copies the localdir directory and its contents to remotedir on the remotehost.

Use a Specific SSH Port

```
scp -P 2222 /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Uses port 2222 for the SSH connection to transfer localfile to the remotehost.

Copy a File and Preserve File Attributes

```
scp -p /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Preserves the file attributes such as modification times when copying localfile.

Limit Bandwidth Usage

```
scp -l 1000 /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Limits the bandwidth used for the transfer to 1000 Kbit/s.

Copy Files with Verbose Output

```
scp -v /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Provides detailed debugging information during the transfer process.

Copy Multiple Files to a Remote Directory

```
scp file1 file2 file3 user@remotehost:/path/to/remotedir
```

Explanation: Copies file1, file2, and file3 to the remotedir on the remotehost.

Use a Different SSH Key for Authentication

```
scp -i /path/to/privatekey /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Uses the specified private key for SSH authentication instead of the default key.

Copy a File to Multiple Remote Hosts

```
scp /path/to/localfile user@host1:/path/to/remotefile user@host2:/path/to/remotefile
```

Explanation: Copies localfile to the same path on host1 and host2.

Copy a File from a Remote Host to Another Remote Host

```
scp user@host1:/path/to/remotefile user@host2:/path/to/destination
```

Explanation: Copies remotefile from host1 to host2. Requires SSH access between the remote hosts.

Copy a File and Show Progress

```
scp -v /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Displays the progress of the file transfer in verbose mode.

Overwrite Remote File without Prompting

```
scp -o StrictHostKeyChecking=no /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Disables host key checking and overwrites remotefile on the remotehost.

Copy File Using IPv4 or IPv6

```
scp -4 /path/to/localfile user@remotehost:/path/to/remotefile
scp -6 /path/to/localfile user@remotehost:/path/to/remotefile
```

Explanation: Forces the use of IPv4 (-4) or IPv6 (-6) for the connection.

36. curl

Introduction: Transfers data from or to a server using various protocols.

Options:

-O: Save file with the same name as on the server.

-L: Follow redirects.

Examples:

1. `curl http://example.com` - Fetch content from URL.
2. `curl -O http://example.com/file.txt` - Download file with original name.
3. `curl -L http://example.com/redirect` - Follow redirects.
4. `curl -u user:password http://example.com` - Access with authentication.
5. `curl -d "param=value" http://example.com` - POST data to URL.
6. `curl -I http://example.com` - Fetch HTTP headers only.

Advance Examples:

Check IP and Port Reachability

```
curl -v telnet://192.168.1.1:80
```

Explanation: Tests connectivity to IP 192.168.1.1 on port 80 using the Telnet protocol.

Fetch HTTP Headers Only

```
curl -I http://example.com
```

Explanation: Retrieves only the HTTP headers from http://example.com.

Download a File with Progress Bar

```
curl -O http://example.com/file.zip
```

Explanation: Downloads file.zip from the specified URL and shows a progress bar.

Follow Redirects

```
curl -L http://example.com
```

Explanation: Follows HTTP redirects to reach the final destination.

Send a POST Request with JSON Data

```
curl -X POST -H "Content-Type: application/json" -d '{"key":"value"}' http://example.com/api
```

Explanation: Sends a POST request with JSON data to http://example.com/api.

Set a User-Agent String

```
curl -A "CustomUserAgent/1.0" http://example.com
```

Explanation: Sets a custom User-Agent string for the request.

Limit the Rate of Data Transfer

```
curl --limit-rate 100K -O http://example.com/largefile.zip
```

Explanation: Limits the download rate to 100 KB/s for largefile.zip.

Check SSL/TLS Certificate Information

```
curl -vI https://example.com
```

Explanation: Retrieves the SSL/TLS certificate information from https://example.com in verbose mode.

Download Multiple Files Concurrently

```
curl -O http://example.com/file1.zip -O http://example.com/file2.zip
```

Explanation: Downloads file1.zip and file2.zip simultaneously.

Use a Proxy Server for the Request

```
curl -x http://proxyserver:8080 http://example.com
```

Explanation: Routes the request through a proxy server at proxyserver on port 8080.

37. useradd

Introduction: Adds a new user to the system.

Options:

-m: Create home directory.

-s: Specify login shell.

Examples:

1. `useradd username` - Add a new user.
2. `useradd -m username` - Add user with a home directory.
3. `useradd -s /bin/bash username` - Specify login shell.
4. `useradd -u 1001 username` - Specify user ID.
5. `useradd -g groupname username` - Specify primary group.
6. `useradd -p password username` - Set user password (encrypted).

38. usermod

Introduction: Modifies an existing user account.

Options:

-aG: Add user to groups.

-s: Change login shell.

Examples:

1. usermod -aG groupname username - Add user to a group.
2. usermod -s /bin/zsh username - Change login shell.
3. usermod -L username - Lock user account.
4. usermod -U username - Unlock user account.
5. usermod -e 2024-12-31 username - Set account expiration date.
6. usermod -c "User Description" username - Change user description.

39. passwd

Introduction: Changes user passwords.

Options: No options.

Examples:

1. passwd - Change the password for the current user.
2. passwd username - Change the password for a specific user.
3. passwd -d username - Delete the password for a user (disable password).
4. passwd -l username - Lock user password.
5. passwd -u username - Unlock user password.
6. passwd -e username - Expire user password (force change on next login).

40. userdel

Introduction: Deletes a user account from the system.

Options:

-r: Remove the user's home directory and mail spool.

Examples:

1. userdel username - Delete a user without removing home directory.
2. userdel -r username - Delete a user and remove home directory.
3. userdel -f username - Force delete a user, even if they are logged in.
4. userdel -r username - Remove user along with their home directory.
5. userdel username -d /home/username - Specify directory to delete.
6. userdel -d /home/username username - Ensure the specific home directory is deleted.

41. groupadd

Introduction: Creates a new group.

Options:

-g: Specify the group ID.

-r: Create a system group.

Examples:

groupadd groupname - Create a new group.

groupadd -g 1001 groupname - Specify the group ID.

groupadd -r groupname - Create a system group.

groupadd -p password groupname - Set a group password.

groupadd -f groupname - Force creation if the group already exists.

groupadd groupname -g 5000 - Create a group with a specific GID.

42. usermod

Introduction: Modifies an existing user account.

Options:

-aG: Add user to groups.

-s: Change login shell.

Examples:

1. usermod -aG groupname username - Add user to a group.
2. usermod -s /bin/zsh username - Change login shell.
3. usermod -L username - Lock user account.
4. usermod -U username - Unlock user account.
5. usermod -e 2024-12-31 username - Set account expiration date.
6. usermod -c "User Description" username - Change user description.

43. cut

Introduction: Removes sections from each line of files.

Options:

-d: Specify delimiter.

-f: Specify fields.

Examples:

1. cut -d: -f1 /etc/passwd - Extract first field using : as delimiter.
2. cut -c1-5 file.txt - Cut characters from 1 to 5.

3. `cut -d, -f2 file.csv` - Extract second field using , as delimiter.
4. `cut -d' ' -f1,3 file.txt` - Extract first and third fields.
5. `cut -d'|' -f1-3 file.txt` - Extract first to third fields.
6. `cut -f2 --output-delimiter=';' file.txt` - Change output delimiter.

Advance Examples:

Extract Multiple Fields

```
cut -d, -f1,3 file.csv
```

Explanation: Extracts the first and third fields from a CSV file using a comma as the delimiter.

Extract a Range of Characters

```
cut -c5-10 file.txt
```

Explanation: Extracts characters from positions 5 to 10 in each line of file.txt.

Extract Fields with a Different Delimiter

```
cut -d: -f2,4 file.txt
```

Explanation: Extracts the second and fourth fields from a file where fields are delimited by colons.

Extract the Last Field in a Delimited File

```
cut -d, -f$(awk -F, '{print NF}' file.csv | head -1) file.csv
```

Explanation: Extracts the last field from a CSV file dynamically by determining the number of fields.

Extract Fields with Multiple Delimiters

```
cut -d' ' -f1,3 --output-delimiter='|' file.txt
```

Explanation: Extracts the first and third fields, with spaces as delimiters, and outputs the result separated by a pipe (|).

Extract Characters and Fields Simultaneously

```
cut -c1-5 --output-delimiter='|' file.txt | cut -d'|' -f2
```

Explanation: Extracts characters 1 to 5 and then extracts the second field from the result.

Extract Fields from a File and Sort the Output

```
cut -d, -f2 file.csv | sort | uniq
```

Explanation: Extracts the second field from a CSV file, sorts the values, and removes duplicates.

Extract and Count Unique Values in a Field

```
cut -d' ' -f2 file.txt | sort | uniq -c
```

Explanation: Extracts the second field from a space-delimited file, sorts it, and counts unique occurrences.

Combine cut with grep to Filter Specific Lines

```
grep 'pattern' file.txt | cut -d' ' -f1,3
```

Explanation: Filters lines containing pattern and extracts the first and third fields.

Extract Field from a File with Variable Delimiters

```
cut -d"$DELIM" -f1 file.txt
```

Explanation: Extracts the first field using a delimiter specified by the environment variable DELIM.

44. sed

Introduction: Stream editor used for parsing and transforming text.

Options:

-e: Specify script to execute.

-i: Edit files in place.

Examples:

1. `sed 's/old/new/' file.txt` - Replace old with new in the file.
2. `sed -i 's/old/new/g' file.txt` - Replace all occurrences in the file.
3. `sed -n '1,5p' file.txt` - Print lines 1 to 5.
4. `sed 's/^[ \t]*//' file.txt` - Remove leading whitespace.
5. `sed -e 's/foo/bar/' -e 's/baz/qux/' file.txt` - Apply multiple expressions.
6. `sed '/pattern/d' file.txt` - Delete lines containing pattern.

Advance Examples:

Replace All Occurrences of a String in a File

```
sed 's/oldstring/newstring/g' file.txt
```

Explanation: Replaces all occurrences of oldstring with newstring in file.txt.

Replace Only the First Occurrence in Each Line

```
sed 's/oldstring/newstring/' file.txt
```

Explanation: Replaces only the first occurrence of oldstring with newstring in each line.

Delete Lines Containing a Specific String

```
sed '/pattern/d' file.txt
```

Explanation: Deletes all lines that contain the pattern.

Insert a Line Before a Specific Line Number

```
sed '3i\This is the inserted line' file.txt
```

Explanation: Inserts This is the inserted line before line 3 in file.txt.

Append a Line After a Specific Line Number

```
sed '3a\This is the appended line' file.txt
```

Explanation: Appends This is the appended line after line 3 in file.txt.

Replace a String in a Specific Line Number

```
sed '3s/oldstring/newstring/' file.txt
```

Explanation: Replaces oldstring with newstring only in line 3 of file.txt.

Delete a Range of Lines

```
sed '5,10d' file.txt
```

Explanation: Deletes lines 5 through 10 from file.txt.

Replace Multiple Strings Using Multiple Commands

```
sed -e 's/old1/new1/g' -e 's/old2/new2/g' file.txt
```

Explanation: Replaces old1 with new1 and old2 with new2 in file.txt.

Print Only Lines Matching a Pattern

```
sed -n '/pattern/p' file.txt
```

Explanation: Prints only the lines that match the pattern.

Remove Leading and Trailing Whitespace from Each Line

```
sed 's/^[ \t]*//;s/[ \t]*$//' file.txt
```

Explanation: Removes leading and trailing whitespace from each line.

Change Case of a String

```
sed 's/[a-z]/\U&/g' file.txt
```

Explanation: Converts all lowercase letters to uppercase in file.txt.

Replace Text with Variable Content

```
sed "s/PLACEHOLDER/$VARIABLE/g" file.txt
```

Explanation: Replaces PLACEHOLDER with the content of the shell variable $VARIABLE.

Extract Text Between Two Patterns

```
sed -n '/start_pattern/,/end_pattern/p' file.txt
```

Explanation: Prints text between start_pattern and end_pattern inclusively.

Replace Text Only in Lines Matching a Pattern

```
sed '/pattern/s/oldstring/newstring/g' file.txt
```

Explanation: Replaces oldstring with newstring only in lines containing pattern.

Remove Empty Lines

```
sed '/^$/d' file.txt
```

Explanation: Deletes all empty lines from file.txt.

45. awk

Introduction: Programming language for pattern scanning and processing.

Options:

-f: Specify file containing awk program.

-v: Assign values to variables.

Examples:

1. `awk '{print $1}' file.txt` - Print the first field of each line.
2. `awk -F: '{print $1}' /etc/passwd` - Print the first field using : as delimiter.
3. `awk '{sum += $1} END {print sum}' file.txt` - Sum the values in the first field.
4. `awk '/pattern/ {print $0}' file.txt` - Print lines matching pattern.
5. `awk -v var=10 '{print $1 + var}' file.txt` - Add a variable to each field.
6. `awk '{if ($1 > 10) print $1}' file.txt` - Print values greater than 10.

Advance Examples:

Print Specific Columns from a File

```
awk '{print $1, $3}' file.txt
```

Explanation: Prints the first and third columns of file.txt.

Sum Values in a Column

```
awk '{sum += $2} END {print sum}' file.txt
```

Explanation: Sums up all values in the second column and prints the total.

Average Values in a Column

```
awk '{sum += $2; count++} END {print sum/count}' file.txt
```

Explanation: Calculates the average of values in the second column.

Print Lines Where a Column is Greater Than a Value

```
awk '$2 > 100' file.txt
```

Explanation: Prints lines where the value in the second column is greater than 100.

Find the Maximum Value in a Column

```
awk 'BEGIN {max = 0} $2 > max {max = $2} END {print max}' file.txt
```

Explanation: Finds and prints the maximum value in the second column.

Replace a Field Delimiter

```
awk -F"," '{OFS=":"; print $1, $2}' file.csv
```

Explanation: Uses a comma as the field delimiter for input and a colon for output.

Print Lines Where a Column Matches a Pattern

```
awk '$1 ~ /pattern/' file.txt
```

Explanation: Prints lines where the first column matches the regex pattern.

Print Line Number and Content

```
awk '{print NR, $0}' file.txt
```

Explanation: Prints each line of file.txt preceded by its line number.

Count Occurrences of a Specific Value in a Column

```
awk '$1 == "value" {count++} END {print count}' file.txt
```

Explanation: Counts the number of occurrences of "value" in the first column.

Calculate and Print Cumulative Sum

```
awk '{sum += $2; print sum}' file.txt
```

Explanation: Calculates and prints a cumulative sum of the second column.

Format Output with Padding

```
awk '{printf "%-10s %s\n", $1, $2}' file.txt
```

Explanation: Prints columns with padded formatting, where the first column is left-aligned with 10 characters.

Print Lines from a File with a Specific Field Count

```
awk 'NF == 3' file.txt
```

Explanation: Prints lines that have exactly three fields.

Add a Header to the Output

```
awk 'BEGIN {print "Header1 Header2"} {print $1, $2}' file.txt
```

Explanation: Prints a custom header followed by the columns of file.txt.

Sort and Print Unique Lines Based on a Column

```
awk '{print $2}' file.txt | sort | uniq
```

Explanation: Extracts the second column, sorts it, and prints unique values.

Filter Lines by Date Range

```
awk '$1 >= "2024-01-01" && $1 <= "2024-12-31"' file.txt
```

Explanation: Prints lines where the date in the first column is within the specified range.

46. sort

Introduction: Sorts lines of text files.

Options:

-n: Numeric sort.

-r: Reverse order.

Examples:

sort file.txt - Sort lines alphabetically.

1. `sort -n file.txt` - Sort lines numerically.
2. `sort -r file.txt` - Sort lines in reverse order.
3. `sort -k2 file.txt` - Sort by the second field.
4. `sort -t, -k1 file.csv` - Sort CSV file by the first column.
5. `sort -u file.txt` - Sort and remove duplicate lines.

Advance Examples:

Sort a File in Reverse Order

```
sort -r file.txt
```

Explanation: Sorts the contents of file.txt in reverse (descending) order.

Sort a File Numerically

```
sort -n file.txt
```

Explanation: Sorts the lines of file.txt numerically rather than alphabetically.

Sort by a Specific Column in a File (e.g., Column 2)

```
sort -k 2 file.txt
```

Explanation: Sorts file.txt based on the second column.

Sort and Remove Duplicate Lines

```
sort -u file.txt
```

Explanation: Sorts file.txt and removes any duplicate lines.

Sort by Multiple Columns (e.g., Column 2, then Column 1)

```
sort -k 2,2 -k 1,1 file.txt
```

Explanation: First sorts by the second column and then by the first column.

Sort a File and Write Output to Another File

```
sort file.txt > sorted_file.txt
```

Explanation: Sorts file.txt and saves the output to sorted_file.txt.

Sort by Month and Day (e.g., Date Format MM/DD)

```
sort -t/ -k1,1 -k2,2n file.txt
```

Explanation: Assumes date format MM/DD in file.txt, sorts by month first and then day numerically.

Sort Based on a Custom Delimiter (e.g., Comma)

```
sort -t, -k 2 file.csv
```

Explanation: Sorts file.csv using a comma as the delimiter and sorts based on the second field.

Sort a File with Case-Insensitive Sorting

```
sort -f file.txt
```

Explanation: Sorts file.txt without considering case sensitivity.

Sort and Output Only Unique Values with Case-Insensitive Option

```
sort -fu file.txt
```

Explanation: Sorts file.txt in a case-insensitive manner and removes duplicates.

47. touch

Introduction: Changes file timestamps or creates empty files.

Options:

-c: Do not create any files.

-t: Set the timestamp.

Examples:

1. `touch file.txt` - Update the timestamp of the file or create it.
2. `touch -c file.txt` - Do not create if the file does not exist.
3. `touch -t 202408060830 file.txt` - Set specific timestamp.
4. `touch file1.txt file2.txt` - Create multiple files.
5. `touch -d "2024-08-06 08:30" file.txt` - Set date and time.
6. `touch -a file.txt` - Change the access time only.

Advance Examples:

Create a New File

```
touch newfile.txt
```

Explanation: Creates a new file named newfile.txt if it does not exist; otherwise, updates the timestamp.

Change File Modification Time to Specific Date and Time

```
touch -t 202408060800 file.txt
```

Explanation: Sets the modification time of file.txt to August 6, 2024, 08:00.

Update Access Time Only

```
touch -a file.txt
```

Explanation: Updates only the access time of file.txt to the current time.

Update Modification Time Only

```
touch -m file.txt
```

Explanation: Updates only the modification time of file.txt to the current time.

Create Multiple Files

```
touch file1.txt file2.txt file3.txt
```

Explanation: Creates file1.txt, file2.txt, and file3.txt if they do not exist, or updates their timestamps.

Set File Timestamp to Match Another File

```
touch -r referencefile.txt targetfile.txt
```

Explanation: Sets the timestamp of targetfile.txt to match the timestamp of referencefile.txt.

Set File Timestamp Using a Date String

```
touch -d "2024-08-06 10:00" file.txt
```

Explanation: Sets the modification time of file.txt to August 6, 2024, 10:00, using a date string.

Create a File with Specific Timestamp and Use a Specific Time Zone

```
TZ='America/New_York' touch -d "2024-08-06 08:00" file.txt
```

Explanation: Sets the timestamp of file.txt to August 6, 2024, 08:00 in the New York time zone.

Change Timestamp of All Files in a Directory

```
touch /path/to/directory/*
```

Explanation: Updates the timestamp of all files in /path/to/directory to the current time.

Create a File with a Timestamp in the Future

```
touch -t 202512312359 futurefile.txt
```

Explanation: Creates futurefile.txt with a modification time set to December 31, 2025, 23:59.

48. find

Introduction: Searches for files and directories.

Options:

-name: Specify file name pattern.

-type: Specify file type.

Examples:

1. `find /path -name file.txt` - Find files named file.txt.
2. `find /path -type d -name dirname` - Find directories named dirname.
3. `find /path -type f -mtime -7` - Find files modified in the last 7 days.
4. `find /path -name "*.log" -delete` - Delete log files.
5. `find /path -type f -size +10M` - Find files larger than 10 MB.
6. `find /path -type f -exec chmod 644 {} \;` - Change permissions of files.

Advance Examples:

Find Files Modified in the Last 7 Days

```
find /path/to/search -type f -mtime -7
```

Explanation: Find files modified in the last 7 days.

Find and Delete Files Larger than 100 MB

```
find /path/to/search -type f -size +100M -exec rm -f {} \;
```

Explanation: Find files larger than 100 MB and delete them.

Find Empty Directories

```
find /path/to/search -type d -empty
```

Explanation: Find empty directories.

Find Files by Name Pattern and Print Details

```
find /path/to/search -type f -name "*.log" -exec ls -lh {} \;
```

Explanation: Find files with a .log extension and list their details.

Find Files and Change Permissions

```
find /path/to/search -type f -name "*.sh" -exec chmod 755 {} \;
```

Explanation: Find .sh files and change their permissions to 755.

Find and Compress Files Larger than 50 MB

```
find /path/to/search -type f -size +50M -exec gzip {} \;
```

Explanation: Find files larger than 50 MB and compress them using gzip.

Find and Move Files to Another Directory

```
find /path/to/search -type f -name "*.txt" -exec mv {} /path/to/destination/ \;
```

Explanation: Find .txt files and move them to /path/to/destination/.

Find Files Not Accessed in the Last 30 Days

```
find /path/to/search -type f -atime +30
```

Explanation: Find files that have not been accessed in the last 30 days.

Find Files with Specific Ownership

```
find /path/to/search -type f -user username
```

Explanation: Find files owned by a specific user username.

Find Files with a Specific Extension Modified in the Last 24 Hours

```
find /path/to/search -type f -name "*.conf" -mtime -1
```

Explanation: Find .conf files modified in the last 24 hours.

Find Files by Size and Print Their Path

```
find /path/to/search -type f -size +1G -print
```

Explanation: Find files larger than 1 GB and print their paths.

Find Files and List Their Disk Usage

```
find /path/to/search -type f -exec du -h {} \;
```

Explanation: Find files and list their disk usage in human-readable format.

Find and Execute a Command on Files

```
find /path/to/search -type f -name "*.tmp" -exec sh -c 'echo {} >> /path/to/tmp_files.log' \;
```

Explanation: Find .tmp files and log their paths to /path/to/tmp_files.log.

Find Files by Permission and Change Owner

```
find /path/to/search -type f -perm 644 -exec chown newowner:newgroup {} \;
```

Explanation: Find files with permissions 644 and change their owner to newowner:newgroup.

Find Files with Specific Pattern in Their Content

```
find /path/to/search -type f -exec grep -l "specific_pattern" {} \;
```

Explanation: Find files containing "specific_pattern" and print their paths

49. cron

Introduction: Schedules tasks to run at specified intervals.

Options:

-e: Edit cron jobs.

-l: List cron jobs.

Examples:

1. crontab -e - Edit user's crontab file.
2. crontab -l - List all cron jobs.
3. crontab -r - Remove the crontab file.
4. crontab -u username -l - List cron jobs for a specific user.
5. echo "0 5 * * * /path/to/command" | crontab - - Add a cron job.
6. crontab -l | grep 'command' - Search for a command in the crontab.

Advance Examples:

Run a Script Every Day at Midnight

```
0 0 * * * /path/to/script.sh
```

Explanation: Executes /path/to/script.sh every day at midnight.

Run a Command Every 15 Minutes

```
*/15 * * * * /path/to/command
```

Explanation: Executes /path/to/command every 15 minutes.

Run a Backup Script Every Sunday at 2 AM

```
0 2 * * 0 /path/to/backup.sh
```

Explanation: Executes /path/to/backup.sh every Sunday at 2 AM.

Run a Script on the First Day of Every Month at 5 AM

```
0 5 1 * * /path/to/monthly_script.sh
```

Explanation: Executes /path/to/monthly_script.sh on the first day of every month at 5 AM.

Run a Command Every Hour Between 8 AM and 6 PM

```
0 8-18 * * * /path/to/hourly_command
```

Explanation: Executes /path/to/hourly_command every hour between 8 AM and 6 PM.

Run a Script on Weekdays at 11 PM

```
0 23 * * 1-5 /path/to/weekday_script.sh
```

Explanation: Executes /path/to/weekday_script.sh every weekday (Monday to Friday) at 11 PM.

Run a Script Every 10 Minutes During Business Hours (9 AM - 5 PM)

```
*/10 9-17 * * * /path/to/business_hours_script.sh
```

Explanation: Executes /path/to/business_hours_script.sh every 10 minutes between 9 AM and 5 PM.

Send a System Report Every Day at 6 PM and Save Output to a File

```
0 18 * * * /usr/bin/system_report.sh >> /var/log/system_report.log 2>&1
```

Explanation: Executes /usr/bin/system_report.sh daily at 6 PM and appends the output to /var/log/system_report.log.

Run a Script Every 3 Hours

```
0 */3 * * * /path/to/every_3_hours.sh
```

Explanation: Executes /path/to/every_3_hours.sh every 3 hours.

Run a Cleanup Script Every Day at 1:30 AM and Redirect Output to a Log File

```
30 1 * * * /path/to/cleanup.sh > /var/log/cleanup.log 2>&1
```

Explanation: Executes /path/to/cleanup.sh daily at 1:30 AM and writes the output to /var/log/cleanup.log.

50. less

Introduction: Views the contents of files one screen at a time.

Options:

-N: Show line numbers.

-S: Chop long lines instead of wrapping.

Examples:

1. `less file.txt` - View file contents.
2. `less -N file.txt` - View file with line numbers.
3. `less -S file.txt` - Disable line wrapping.
4. `less +G file.txt` - Start viewing from the end of the file.
5. `less file1.txt file2.txt` - View multiple files.
6. `less -p "pattern" file.txt` - Search for a pattern in the file.

Advance Examples:

Search for a Pattern and View File

```
grep "pattern" file.txt | less
```

Explanation: Search for "pattern" in file.txt and pipe the results to less for paginated viewing.

View Log Files with Filtering

```
tail -f /var/log/syslog | less
```

Explanation: Follow the syslog file in real-time and pipe the output to less for easier viewing.

Combine grep and less for Multiple Files

```
grep -r "pattern" /path/to/directory | less
```

Explanation: Recursively search for "pattern" in a directory and view the results with less.

Use find with less to View Results

```
find /path/to/search -type f -name "*.log" | less
```

Explanation: Find all .log files in a directory and view the list with less.

View less Output of a Compressed File

```
zcat file.gz | less
```

Explanation: Decompress file.gz on the fly and pipe the output to less.

Search and View Output from ps Command

```
ps aux | less
```

Explanation: List all running processes with ps and view the output with less.

View Sorted Output with sort and less

```
sort file.txt | less
```

Explanation: Sort the lines of file.txt and view the sorted result with less.

Combine awk Output with less

```
awk '{print $1}' file.txt | less
```

Explanation: Extract the first field of file.txt using awk and view the result with less.

Paginate Search Results with less and grep

```
grep "pattern" file1.txt file2.txt | less
```

Explanation: Search for "pattern" in multiple files and view the results with less.

DISCLAIMER AND CONSENT