***Commands***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***ls*** | List directory contents | ***man*** | User manul | ***top*** | Display linuc tasks |
| ***cd*** | Change the current directory | ***Alias*** | Shortcuts for commands | ***df*** | File system disk space usage |
| ***pwd*** | Print the current working directory | ***grep*** | Search Text Using Patterns | ***du*** | File space usage |
| ***echo*** | Display a line of text | ***awk*** | Pattern Scanning and Processing Language | ***free*** | Display free and used memory |
| ***cat*** | Concatenate and display files | ***sed*** | Stream Editor | ***kill*** | Terminate process |
| ***cp*** | Copy files and directories | ***cut*** | Remove section from lines | ***zip*** | Package and compress (archive) files |
| ***mv*** | Move or rename files | ***sort*** | Sort lines of text files | ***unzip*** | Extract from ZIP archive |
| ***rm*** | Delete files or folders | ***tail*** | Display last part of files | ***tar*** | used to create, maintain, modify, and extract files from an archive file. |
| ***touch*** | Create an empty file or update its time | ***head*** | Display the beginning of a file | ***chmod*** | Change File Permissions |
| ***mkdir*** | Create a new folder | ***ps*** | Snapshot of current processes | ***chown*** | change the ownership of files and directories |
|  |  |  |  | ***chgrp*** | Change Group Ownership |

|  |  |  |
| --- | --- | --- |
| ***ls*** | **ls- List directory contents** | command is used to list the contents of a directory.can display files, directories, and information about them. |
| ***ls -l*** | **-l - Long listing format** | it gives you detailed information about files and folders. |
| ***ls -a*** | **-a - Include hidden files** | is helpful when you need to view or manage configuration files that are not visible by default. |
| ***ls -lh*** | **-h - Human-readable sizes** | makes file sizes easier to read.his option is particularly useful when you want to quickly assess the size of files and directories without manually converting bytes. |
| ***ls -t*** | **-t - Sort by modification time** | sorts files and directories by modification time, with the most recently modified files first. This option is useful when you want to see the most recently updated files first. |
| ***ls -r*** | **-r - Reverse order while sorting** | When used in combination with other options like -t, it can display the oldest files first.  This option is useful for reversing the default sorting behavior to meet specific needs. |
| ***ls -R*** | **-R -List subdirectories recursively** | lists directories and their contents recursively. This is useful for viewing the entire directory tree. |
| ***ls -S*** | **-S - Sort by file size** | option sorts files by size, with the largest files first.This is helpful for quickly identifying large files in a directory. |
| ***ls -1*** | **-1 - List one file per line** | option lists one file per line, which is useful for scripts or when piping output to other commands. |
| ***ls -d \*/*** | **-d - List directories themselves, not their contents** | option lists directories themselves rather than their contents. |
| ***ls -F*** | **-F-Append indicator (one of \*/=@|) to entries** | option appends an indicator character to entries (e.g., / for directories, \* for executables). |
| ***ls -l -a*** | **Using Multiple Options- *ls-la, ls-ltr*** | ls -l -a will display a detailed listing of all files and directories, including hidden files. |

|  |  |  |  |
| --- | --- | --- | --- |
| ***cd*** | cd directory\_name | **Cd- Change the current directory** | used to change the current working directory in the terminal. |
| ***cd ..*** |  | **cd .. Move up one directory level** | command lets you go to the folder above your current one. It's useful when you need to go to the parent folder. |
| ***cd~*** |  | **cd ~: Change to the home directory** | command takes you to your home directory, which is the default directory for your user account. This option is useful when you need to return to your starting point after navigating through various directories. |
| ***cd-*** |  | **cd -: Switch to the previous directory** | command switches your working directory to the previous one you were in.  This option is useful for toggling between two directories without needing to type their full paths repeatedly. |
| ***cd /*** |  | **cd /: Change to the root directory** | command takes you to the root directory of the file system.  This option is useful when you need to access system-wide files or directories. |
|  | cd my\_directory && ls |  | using cd with ls can quickly show you the contents of a directory you navigate to: |

|  |  |  |
| --- | --- | --- |
| ***pwd*** | **Print the current working directory** | The pwd command shows you the full path of the folder you're currently in. |
| ***pwd -L*** | -L: Display the logical current working directory | The -L option shows the logical path, including any symbolic links. |
| ***pwd -P*** | -P: Display the physical current working directory (without symbolic links) | The -P option shows the physical path, resolving any symbolic links to their actual locations. This is useful when you need to know the exact physical directory structure. |

|  |  |  |
| --- | --- | --- |
| ***Echo - Display a line of text*** | echo "message" : echo "Hello, World!" | command is used to show a line of text or a variable's value in the terminal. |
| *-n - Don't add a new line at the end* | echo -n "Hello,";echo " World!" | option prevents echo from adding a newline at the end of the output. This is useful when you want to continue output on the same line. |
| *-e - Allow special characters like \n for new lines* | echo -e "Hello\nWorld!"  Output = Hello  World! | option enables the use of backslash escapes like \n for new lines, \t for tabs, etc. |
| *-E - Don't allow special characters* | echo -E "Hello\nWorld!" | option disables the use of backslash escapes default. |

|  |  |  |
| --- | --- | --- |
| ***Cat -Concatenate and display files*** | cat filename:  Ex : cat my\_file.txt | The cat command is used to show the content of files in the terminal.  You can also use it to combine multiple files into one |
| *-n  Add numbers to each line* | cat -n my\_file.txt | The -n option adds numbers to each line of the output. |
| *-b - Add numbers only to lines with text* | cat -b my\_file.txt | The -b option adds numbers only to lines with text, ignoring blank lines. |
| *-s - Remove extra empty lines* | cat -s my\_file.txt | The -s option removes extra empty lines from the output, leaving only a single blank line where multiple ones existed. |
| *-v - Show non-printing characters (except for tabs and end of line)* | cat -v my\_file.txt | The -v option makes non-printing characters visible, except for tabs and end-of-line characters.  This is useful for debugging files with hidden characters. |
| *-E -show $ at the end of line* |  |  |
| *>* | cat>file1.txt | > used to create a new file and enter text contents from the terminal |
| *>>* | cat>> file1.txt  Ex : cat file1.txt>>file2.txt | >> use to append text contents to an existing file from the terminal |
| *Concatenate Two Files* | cat file1.txt file2.txt > combined.txt | The cat command can be used to concatenate multiple files into one.  This is useful for combining files or appending content to an existing file.  This command takes the contents of file1.txt and file2.txt and writes them into combined.txt. |
| *Using cat with Piping* | cat my\_file.txt | grep "shells" | The cat command is often used with piping to send the content of files to other commands. This is useful for processing text data. |

|  |  |  |
| --- | --- | --- |
| **Cp - Copy files and directories** | cp source\_file destination\_file  Ex: cp my\_file.txt copy\_of\_my\_file.txt | The cp command is used to copy files and directories from one location to another. It's like making a duplicate of your file or folder. |
| *-r - Copy all files and folders inside a directory* | cp -r images images2 | The -r option lets you copy entire directories, including all files and subdirectories. |
| *-i - Ask before replacing files* | cp -i my\_file.txt copy\_of\_my\_file.txt | The -i option will prompt you before overwriting files, helping you avoid accidental replacements. |
| *-u - Copy only if the source is newer* | cp -u new\_file.txt existing\_file.txt | The -u option copies files only if the source file is newer than the destination file. |
| *-v - Verbose mode, show files being copied* | cp -v my\_file.txt copy\_of\_my\_file.txt | The -v option enables verbose mode, which displays the files being copied in the terminal.  This is useful for tracking the copy process, especially when dealing with many files. |
| *Using cp with Wildcards* | cp \*.txt /destination/ | Wildcards allow you to copy multiple files at once. For example, cp \*.txt /destination/ will copy all text files to the destination folder. |

|  |  |  |
| --- | --- | --- |
| *Mv - Move or Rename Files* | To move a file,  **mv source\_file destination\_directory:**  Ex: mv my\_file.txt /path/to/destination/  To rename a file : mv old\_name.txt new\_name.txt | The mv command is used to move or rename files and directories.  It's like changing where a file is or what it's called. |
| *-i - Ask before replacing files* | **mv -i my\_file.txt myfolder/**  mv: overwrite 'myfolder/my\_file.txt'? | The -i option will prompt you before overwriting files, helping you avoid accidental replacements. |
| *-u - Move only if the source is newer* | **mv -u new\_file.txt /path/to/destination/** | he -u option moves files only if the source file is newer than the destination file. |
| *-v - Verbose mode, show files being moved* | **mv -v my\_file.txt myfolder/new\_directory/** renamed 'my\_file.txt' -> 'myfolder/new\_directory/my\_file.txt' | The -v option enables verbose mode, which displays the files being moved in the terminal.  This is useful for tracking the move process, especially when dealing with many files. |
| *Using mv with Wildcards* | mv \*.txt /destination/ | Wildcards allow you to move multiple files at once. For example, mv \*.txt /destination/ will move all text files to the destination folder. |

|  |  |  |
| --- | --- | --- |
| ***Rm - Remove Files or Directories*** | rm filename  Ex : rm my\_file.txt | The rm command is used to remove files or directories. as removed files cannot be easily recovered |
| *-r - Delete a folder and everything inside it* | rm -r directory | The -r option allows you to delete directories and all their contents. |
| *-i - Ask before deleting each file* | rm -i my\_file.txt  rm: remove regular file 'my\_file.txt'? | The -i option will prompt you before each file is deleted, helping you avoid accidental deletions. |
| *-f - Force delete without asking* | rm -f file.txt | The -f option forces deletion without any prompts. |
| *-v - Verbose mode, show files being removed* | **rm -v my\_new\_file.txt**  removed 'my\_new\_file.txt' | The -v option enables verbose mode, which displays the files being removed in the terminal.  This is useful for tracking the removal process, especially when dealing with many files. |
| *Using rm Safely* | rm -i -r images | Always double-check the files and directories you are about to delete, especially when using the -r and -f options. |

|  |  |  |
| --- | --- | --- |
| ***Touch - Change File Timestamps or create an empty file*** | touch filename  Ex : touch file.txt | The touch command is used to change file timestamps or create an empty file if it doesn't exist.  It's often used to create placeholder files or update timestamps for build systems. |
| *-a - Update only when the file was last read* | touch -a file.txt | The -a option lets you update the access time of a file. |
| *-m - Update only when the file was last changed* | touch -m my\_file.txt | The -m option lets you update only the modification time of a file. |
| *-t - Set the timestamp to a specific time* | touch -t 202501010000 my\_file.txt | The -t option allows you to set the timestamp to a specific time. |
| *-c - Do not create any files* | touch -c non\_existent\_file.txt | The -c option tells touch not to create any files if they do not exist.  This is useful when you want to update timestamps without accidentally creating new files. |
| *Using touch with Wildcards* | touch \*.txt  will update the timestamps of all text files in the directory. | Wildcards allow you to update timestamps on multiple files at once. |

|  |  |  |
| --- | --- | --- |
| **Mkdir - Make Directories** | mkdir directory\_name  Ex : mkdir new\_directory | The mkdir command is used to create new directories. It's a simple way to organize files into folders. |
| *-p - Create parent directories as needed* | mkdir -p parent/child |  |
| *-v - Show a message for each created directory* | mkdir -v new\_directory | The -v option prints a message for each directory created, which is helpful for tracking what happens. |
| *-m - Set file mode (permissions)* | mkdir -m 755 new\_directory | The -m option allows you to set the file mode (permissions) for the new directories.  This is useful for ensuring the correct permissions are set at creation. |

|  |  |  |
| --- | --- | --- |
| *Man - User Manual* | Syntax : **man [command]** Ex : man ls  man grep | The man command is used to display the user manual of any command that can be run on the terminal.  It's a valuable resource for understanding command usage, options, and examples. |

|  |  |  |
| --- | --- | --- |
| ***alias*** | syntax ***alias name='command'***  Where,  name is the shortcut you want to use   command is the full command you want to run.  Ex : ***alias ll='ls -la' , alias gs='git status'*** | Aliases in Bash allow you to create shortcuts for long or frequently used commands. This makes it easier to execute complex commands with a simple keyword. |
| ***unalias*** | ***unalias gs*** | remove an alias |

|  |  |  |
| --- | --- | --- |
| ***Grep - Search Text Using Patterns*** | Syntax : ***grep 'pattern' filename***  ***Ex :*** grep 'shell' file.txt | The grep command is used to search for text patterns within files.  It's a powerful way to find specific text in large files or across many files. |
| *-i - Search ignoring case differences*  *(uppercase or lowercase)* | grep -i 'shell' file.txt | The -i option lets you search without worrying about case sensitivity. |
| *-r - Recursive Search* | grep -r 'search\_term' /home/user/my\_directory | Search through all files in a directory and its subdirectories |
| *-v - Invert Match* | grep -v 'shell' my\_file.txt | Find lines that do not match the pattern |
| *Using grep with Regular Expressions* inds lines starting with a letter. | grep '^[A-Za-z]' my\_file.txt | Regular expressions allow you to search for complex patterns. |

|  |  |  |
| --- | --- | --- |
| *Awk - Pattern Scanning* *and Processing Language* |  | The awk command is used for pattern scanning and processing language.  It's useful for handling text files and used for data extraction and reporting. |
| *-F - Field Separator* | awk -F"," '{print $1}' example\_data.csv | The -F option allows you to define the field separator for processing data. This is useful when dealing with CSV files or data with specific delimiters. |
| *-v - Assign Variable* | awk -v var="Amount:" '{print var, $3}' example\_data.csv |  |
| *Using awk for Data Manipulation* | awk '{sum += $3} END {print sum}' example\_data.csv | calculates the sum of the Amount column. |
| *Common Errors and Troubleshooting* | When using awk, you might encounter errors such as:  "awk: syntax error" - Check your command syntax.  "awk: cannot open file" - Ensure the file path is correct and accessible. | Debugging tips include using print statements to check variable values and logic flow. |

|  |  |  |
| --- | --- | --- |
| *Cut - Remove Sections from Lines* | To extract the first field of a file  cut -f1 filename  Ex: cut -f1 example\_data.txt | The cut command is used to remove sections from each line of files.  It's a useful tool for extracting specific fields of data from a file or output stream. |
| *-d - Choose what separates the fields* *Specify a Delimiter* | cut -d',' -f1 example\_data.txt |  |
| *-f - Select specific fields to display* | cut -f1-2 example\_data.txt |  |
| *--complement - Show all fields except the selected ones* | cut --complement -f1 example\_data.txt |  |
| *Advanced Field Extraction* | cut -d' ' -f2-3 example\_data.txt  extracts fields 2 through 3 from the file. |  |

|  |  |  |
| --- | --- | --- |
| Sed -Command - Stream Editor | To replace the first occurrence of a pattern in a file  sed 's/old/new/' filename  Ex: sed 's/World/Bash/' example\_text.txt | The sed command is a stream editor used to perform basic text transformations on an input stream (a file or input from a pipeline).  It's a powerful tool for making quick edits to files or streams of data. |
| *-i - Edit files directly without needing to save separately.* *Edit Files In Place* | sed -i 's/World/Bash/g' example\_text.txt | The -i option allows you to edit files directly without needing to save separately.  Without this option, sed outputs the result to the standard output, and you must redirect it to a file to save changes. |
| *-e - Add the script to the commands to be executed* |  |  |
| *-n - Don't automatically print lines* *Suppress Printing* | sed -n 's/World/Bash/p' example\_text.txt | The -n option suppresses automatic printing of pattern space.  By default, sed prints each line of input to the output. Using -n allows you to control which lines are printed, typically with the p command. |
| *-r - Use extended regular expressions* | sed -r 's/(World|Line)/Hello/g' example\_text.txt | The -r option allows the use of extended regular expressions, which provide more powerful pattern matching capabilities than basic regular expressions.  Without this option, sed uses basic regular expressions. |
| *-f - Add script from a file* | sed -f script.sed example\_text.txt | The -f option allows you to add a script from a file, which is useful for executing complex or multiple sed commands.  Without this option, you must specify the script directly in the command line.  Content o**f script.sed file:**  **s/World/Bash/g** |
| *-l -Specify line length for l command* | sed -l 10 'l' example\_text.txt  This option appends a $ at the end of each line to indicate the end of the line. | The -l option specifies the line length for the l command, which prints lines with non-printable characters.  This option is useful for formatting output when dealing with long lines. |
| *Redirect Output to a File* | sed 's/World/Bash/' example\_text.txt > new\_example\_text.txt | To save the changes made by sed to a file, you can redirect the output to a new file. This is useful when you don't want to overwrite the original file. |
| *Using sed for Advanced Text Processing* | Sed can perform advanced text processing tasks.  sed 's/^/Prefix: /' example\_text.txt |  |
| *Common Errors and Troubleshooting* | * sed: command garbled" - Check your command syntax. * "sed: can't read file" - Ensure the file path is correct and accessible. | Debugging tips include using echo to print intermediate results and verify command logic. |

|  |  |  |
| --- | --- | --- |
| Sort -Sort Lines of Text Files | sort filename  Ex : sort fruits.txt | The sort command is used to sort lines of text files.  It's a handy tool for organizing data in files. |
| -r - Sort in reverse order | sort -r fruits.txt |  |
| -n - Sort numbers correctly | sort -t "," -n -k2,2 fruits.txt | The -n option allows you to sort numbers correctly.  Without this option, numbers are sorted lexicographically, meaning "10" would come before "2". |
| -k - Sort by a specific column | sort -t "," -k2,2 fruits.txt | The -k option allows you to sort by a specific column.  Without this option, sort uses the entire line as the key. |
| -u - Remove duplicate lines | sort -u fruits.txt | The -u option removes duplicate lines from the output. Without this option, duplicate lines are retained. |
| -t - Specify a delimiter for fields | sort -t "," -k2,2 fruits.txt | The -t option specifies a delimiter for fields, which is useful for sorting files with a specific field separator.  Without this option, sort assumes whitespace as the default delimiter. |
| Complex Sorting | sort -t "," -k1,1 -k2,2r fruits.txt | sorts by the first column, and then by the second column in reverse order. |

|  |  |  |
| --- | --- | --- |
| Tail - Display Last Part of Files | syntax of the tail command is  tail [OPTION]... [FILE]…  Ex : tail logfile.txt | The tail command is used to display the last part of files.  It's particularly useful for viewing the end of log files or any file that is being updated in real-time. |
| *-n [number]: Display the last [number] lines of the file.* | tail -n 5 logfile.txt | The -n option allows you to specify the number of lines to display from the end of the file.  By default, tail shows the last 10 lines. |
| *-f: Follow the file as it grows, useful for monitoring log files.* | tail -f logfile.txt | The -f option is used to follow a file as it grows, which is particularly useful for monitoring log files in real-time. |
| *-c [number]: Display the last [number] bytes of the file.* | tail -c 20 logfile.txt | The -c option allows you to display the last [number] bytes of a file instead of lines. |
| *--pid=[pid]: Terminate after the process with the given PID dies.* | tail -f --pid=1234 logfile.txt | The --pid option terminates tailing after the process with the given PID dies. This is useful for stopping the tail operation when a related process ends. |
| *--retry: Keep trying to open a file even if it is inaccessible.* | tail --retry -f logfile.txt | The --retry option makes tail keep trying to open a file even if it is inaccessible. This is useful for files that may be temporarily unavailable. |

|  |  |  |
| --- | --- | --- |
| Head - Display the beginning of a file | head filename.txt  Ex : head logfile.txt | The head command is used to display the first part of files.  It's particularly useful for previewing the start of a file to understand its structure. |
| -n [number]: Display the first [number] lines of the file. | head -n 5 logfile.txt | The -n option allows you to specify the number of lines to display from the start of the file |
| -c [number]: Display the first [number] bytes of the file. | Ex : Display First 20 Bytes head -c 20 logfile.txt | The -c option allows you to display the first [number] bytes of a file instead of lines. |
| Multiple Files | Ex :  Display First 3 Lines of Multiple Files ead -n 3 logfile.txt logfile2.txt | The head command can be used to display the beginning of multiple files. By default, it prints the file name as a header before the content of each file. |
| Option: -q Suppress Headers | head -q -n 3 logfile.txt logfile2.txt | The -q option suppresses the printing of headers when multiple files are being processed. This is useful when you want to view the contents of multiple files without the file names being printed. |

|  |  |
| --- | --- |
| ps - Snapshot of Current Processes | **ps**  The ps command is used to report a snapshot of current processes.  It's a useful tool for monitoring and managing processes on your system. |
| -e - Show all processes | ps -e |
| -f - Show detailed information | ps -f |
| -u - Show processes for a specific user | ps -u user |
| -a - Show all processes with a terminal | ps -a |
| -x - Show processes without a terminal | ps -x |
| Combining Options | Options can be combined to provide more detailed output. For example, ps -ef shows all processes with detailed information. |

|  |  |  |
| --- | --- | --- |
| Top - Display Linux Tasks | top | The top command is used to display Linux tasks.  It's a powerful tool for monitoring system performance in real-time. |
| -d - Set the time between updates | top -d 5 | The -d option allows you to set the time between updates. |
| -p - Monitor specific PIDs | top -p 1234 | The -p option allows you to monitor specific PIDs. |
| -u - Show tasks for a specific user | top -u user | The -u option allows you to show tasks for a specific user. |
| -n - Set the number of iterations | top -n 2 | The -n option allows you to set the number of iterations before top exits. |
| -b - Batch mode operation | top -b -n 1 | The -b option allows top to run in batch mode, suitable for sending output to other programs or files. |
| Combining Options | top -b -n 1 | Options can be combined to provide more detailed output. For example, top -b -n 1 runs top in batch mode for one iteration. |

|  |  |  |
| --- | --- | --- |
| Df - File System Disk Space Usage | df | The df command is used to report file system disk space usage.  It's a useful tool for checking available storage on your system. |
| -h - Show sizes in human-readable format (e.g., KB, MB) | df -h | The -h option allows you to show sizes in human-readable format. |
| -a - Show all file systems, even empty ones | df -a | The -a option allows you to show all file systems, even empty ones. |
| -T - Show the type of file system | df -T |  |
| -i - Show inode usage | df -i | ****Inodes:**** **Inodes are data structures used by many file systems to store information about files and directories, such as their size, owner, permissions, and timestamps.**  Each file or directory has a unique inode. The df -i command shows inode usage, which can be important for systems with many small files. |
| * -P - Use POSIX output format | df -P | ****POSIX:** POSIX (Portable Operating System Interface) is a set of standards specified by the IEEE for maintaining compatibility between operating systems.**  The df -P option provides output in a POSIX-compliant format, ensuring consistency across different environments and systems. |
| Combining Options | df -hT | Optons can be combined to provide more detailed output. |
| Understanding the Output The df command output consists of several columns, each representing different aspects of the file system's disk usage: |  | * ****Filesystem****: The name of the file system. * ****1K-blocks****: Total size of the file system in 1K blocks. * ****Used****: Amount of space used. * ****Available****: Amount of space available for use. * ****Use%****: Percentage of space used. * ****Mounted on****: Directory where the file system is mounted. |

|  |  |  |
| --- | --- | --- |
| Du - File Space Usage | duUnderstanding the Output The du command output consists of two columns:   * ****Size****: The amount of disk space used by the file or directory. * ****Path****: The file or directory path. | he du command is used to estimate file space usage.  It's helpful for finding out how much space files and directories take up. |
| -h - Show sizes in human-readable format (e.g., KB, MB) | du -h |  |
| -s - Show only the total size for each item | du -s |  |
| -a - Show sizes for all files, not just directories | du -a |  |
| -c - Produce a grand total | du -c |  |
| --max-depth=N - Limit the depth of directory traversal Limit Directory Traversal Depth to 2 | du --max-depth=1  In this example, du --max-depth=1 shows the space used by each directory at the top level, without diving deeper into subdirectories.  du --max-depth=2  Here, du --max-depth=2 provides a summary of disk usage up to two levels deep, including subdirectories. | The --max-depth=N option allows you to limit the depth of directory traversal.  This can be useful for summarizing disk usage at a specific directory level. |
| Combining Options | du -h --max-depth=1  shows sizes in human-readable format with limited directory depth. | Options can be combined to provide more detailed output |

|  |  |  |
| --- | --- | --- |
| Free - Display Free and Used Memory | free | The free command is used to display the amount of free and used memory in the system.  It's useful for monitoring memory usage and managing system resources. |
| -h - Show memory in human-readable format (e.g., KB, MB, GB) | free -h | The -h option allows you to show memory in a human-readable format, such as KB, MB, or GB. |
| -b - Show memory in bytes | free -b | The -b option allows you to show memory in bytes, providing a more precise measurement. |
| -k - Show memory in kilobytes (KB) | free -k | The -k option allows you to show memory in kilobytes.  This is the default behavior of the free command. |
| -m - Show memory in megabytes (MB) | free -m | The -m option allows you to show memory in megabytes, which can be easier to read for larger memory sizes. |
| -g - Show memory in gigabytes (GB) | free -g | The -g option allows you to show memory in gigabytes, which is useful for systems with large amounts of memory. |
| -s [interval] - Continuously display memory usage at specified intervals | free -s 5  This command will update the memory usage every 5 seconds | The -s option allows you to continuously display memory usage at specified intervals. This is useful for monitoring memory usage over time. |
| -t - Display total memory | free -t | The -t option includes a line showing the total memory (used + free) for both RAM and swap. This provides a quick overview of total memory resources. |

|  |  |  |
| --- | --- | --- |
| Kill - Terminate Processes | kill [OPTION]..  The kill command is commonly used to:   * Terminate unresponsive processes. * Manage system resources by stopping unnecessary processes. * Send specific signals to processes for custom handling. | The kill command is used to terminate processes in a Unix-like operating system.  It's a powerful tool for managing system resources and ensuring that processes do not consume more resources than necessary. |
| -9: Forcefully terminate a process. | kill -9 1234 | The -9 option sends the SIGKILL signal to a process, which forcefully terminates it. This is useful when a process does not respond to other signals. |
| -l: List all signal names. | kill -l | The -l option lists all available signal names. This can help you understand which signals are available for use with the kill command. |
| -s [signal]: Specify a signal to send. | kill -s SIGTERM 1234 | The -s option allows you to specify a signal to send to a process. This provides flexibility in controlling processes. |
| -p: Print the process ID. | kill -p 1234 | The -p option prints the process ID of the process you are targeting. This is useful for verifying the process you intend to signal. |

|  |  |  |
| --- | --- | --- |
| Zip - package and compress (archive) files | zip archive.zip file1 file2 | The zip command is used to package and compress files into a ZIP archive. |
| -r - Recursively zip directories | zip -r archive.zip folder/ | The -r option allows you to zip directories and their contents recursively. |
| -u - Update files in the archive if they are newer | zip -u archive.zip file1 file2 | The -u option updates files in the archive only if they are newer than the existing files. |
| -d - Delete files from the archive | zip -d archive.zip file1 | The -d option deletes specified files from the archive. |
| -e - Encrypt the contents of the ZIP archive | zip -e archive.zip file1 file2 | The -e option encrypts the contents of the ZIP archive, requiring a password to unzip. |
| -x - Exclude specific files from being zipped | zip archive.zip file1 file2 -x file2 | The -x option excludes specific files from being added to the archive. |

|  |  |  |
| --- | --- | --- |
| Unzip - Extract from ZIP archive | unzip archive.zip | he unzip command is used to extract compressed files from a ZIP archive. |
| -l - List archive files | unzip -l archive.zip | The -l option lists all the files in the ZIP archive without extracting them. |
| -t - Test compressed archive files | unzip -t archive.zip | The -t option tests the integrity of the files in the ZIP archive. |
| -d - Extract files into a different directory | unzip -d /path/to/extract/ archive.zip | The -d option extracts files to a specified directory. |
| -o - Overwrite existing files without prompting | unzip -o archive.zip | The -o option overwrites existing files without prompting. |
| -x - Exclude specific files from being extracted | unzip archive.zip -x file1 | The -x option excludes specific files from being extracted. |

|  |  |  |
| --- | --- | --- |
| Tar - An archiving utility | Tar | The tar command is used to create, maintain, modify, and extract files from an archive file. |
| -c - Create a new archive | tar -cvf archive.tar file1 file2 |  |
| -x - Extract files from an archive | tar -xvf archive.tar |  |
| -t - List the contents of an archive | tar -tvf archive.tar |  |
| -z- Filter the archive through gzip | tar -czvf archive.tar.gz file1 file2 |  |
| -v - Verbosely list files processed | tar -xvf archive.tar | The -v option provides verbose output, listing files as they are processed. |
| -f - Specify the filename of the archive | tar -cvf archive.tar file1 file2 | The -f option specifies the filename of the archive to create or operate on. It should be used as the last option before the archive name. |

|  |  |  |
| --- | --- | --- |
| Bash File Permissions and Ownership |  | Each file has an owner, a group, and a set of permissions that determine who can read, write, or execute the file. |
| File Permissions : File permissions are represented by a series of characters that indicate the permissions for the owner, the group, and others. The permissions are | * r: Read permission * w: Write permission * x: Execute permission | For example, the permission rwxr-xr-- means the owner can read, write, and execute the file, the group can read and execute, and others can only read. |
| Numeric Representation of Permissions : File permissions can also be represented numerically, which is often used in scripts and command-line operations: | * 0: No permission * 1: Execute permission * 2: Write permission * 3: Write and execute permissions * 4: Read permission * 5: Read and execute permissions * 6: Read and write permissions * 7: Read, write, and execute permissions | For example, the numeric permission 755 means the owner can read, write, and execute (7), and the group and others can read and execute (5). |
| File Ownership : Each file has an owner and a group associated with it. The owner is typically the user who created the file, and the group is a collection of users who share access to the file. | * chmod: Change file permissions * chown: Change file ownership * chgrp: Change group ownership |  |
| Chmod - Change File Permissions | chmod [options] mode[,mode] file1 [file2 ...]  The chmod command is commonly used to:   * Set executable permissions for scripts. * Restrict access to sensitive files. * Ensure that files have the correct permissions for sharing. | The chmod command is used to change the file permissions in Unix-like operating systems.  It allows you to set who can read, write, or execute a file. |
| -R: Change files and directories recursively. | chmod -R 755 /path/to/directory |  |
| -v: Output a diagnostic for every file processed. | chmod -v 644 file.txt |  |
| Chown -Change File Ownership | chown [options] user[:group] file1 [file2 ...] | The chown command is used to change the ownership of files and directories in Unix-like operating systems. |
| -R: Change files and directories recursively. | chown -R user:group /path/to/directory | The -R option allows you to change ownership for files and directories recursively. |
| -v: Output a diagnostic for every file processed. | chown -v user file.txt | The -v option provides verbose output, showing a diagnostic message for each file processed by the command. |
| Chgrp - Change Group Ownership | chgrp [options] group file1 [file2 ...] | The chgrp command is used to change the group ownership of files and directories in Unix-like operating systems. It allows you to set which group owns a file. |
| -R: Change files and directories recursively. | chgrp -R group /path/to/directory | The -R option allows you to change group ownership for files and directories recursively |
| -v: Output a diagnostic for every file processed. | chgrp -v group file.txt | provides verbose output, showing a diagnostic message for each file processed by the command. |