LINUX Command Summary

1. File and Directory Operations Commands

Command	Description	Options	Examples
ls	List files and directories.	• **-l**: Long format listing. • **-a**: Include hidden files hidden ones • **-h**: Human-readable file sizes.	• **ls -l** displays files and directories with detailed information. • **ls -a** shows all files and directories, including • **ls -lh** displays file sizes in a human-readable format.
cd	Change directory.		• **cd /path/to/directory** changes the current directory to the specified path.
pwd	Print current working directory.		• **pwd** displays the current working directory.
mkdir	Create a new directory		• **mkdir my_directory** creates a new directory named “my_directory”.
rm	Remove files and directories.	• **-r**: Remove directories recursively. • **-f**: Force removal without confirmation.	• **rm file.txt** deletes the file named “file.txt”. • **rm -r my_directory** deletes the directory “my_directory” and its contents. • **rm -f file.txt** forcefully deletes the file “file.txt” without confirmation.
cp	Copy files and directories.	• **-r**: Copy directories recursively.	• **cp -r directory destination** copies the directory “directory” and its contents to the specified destination. • **cp file.txt destination** copies the file “file.txt” to the specified destination.
mv	Move/rename files and directories.		• **mv file.txt new_name.txt** renames the file “file.txt” to “new_name.txt”. • **mv file.txt directory** moves the file “file.txt” to the specified directory.
touch	Create an empty file or update file timestamps.		• **touch file.txt** creates an empty file named “file.txt”.
cat	View the contents of a file.		• **cat file.txt** displays the contents of the file “file.txt”.
head	Display the first few lines of a file.	• **-n**: Specify the number of lines to display.	• **head file.txt** shows the first 10 lines of the file “file.txt”. • **head -n 5 file.txt** displays the first 5 lines of the file “file.txt”.

Command	Description	Options	Examples
tail	Display the last few lines of a file.	• **-n**: Specify the number of lines to display.	• **tail file.txt** shows the last 10 lines of the file “file.txt”. • **tail -n 5 file.txt** displays the last 5 lines of the file “file.txt”.
ln	Create links between files.	• **-s**: Create symbolic (soft) links.	• **ln -s source_file link_name** creates a symbolic link named “link_name” pointing to “source_file”.
find	Search for files and directories.	• **-name**: Search by filename. • **-type**: Search by file type.	• **find /path/to/search -name “*.txt”** searches for all files with the extension “.txt” in the specified directory.

2. File Permission Commands

Command	Description	Options	Examples
chmod	Change file permissions.	• **u**: User/owner permissions. • **g**: Group permissions. • **o**: Other permissions. • **+**: Add permissions. • **–**: Remove permissions. • **=**: Set permissions explicitly.	• **chmod u+rwx file.txt** grants read, write, and execute permissions to the owner of the file.
chown	Change file ownership.		• **chown user file.txt** changes the owner of “file.txt” to the specified user.
chgrp	Change group ownership.		• **chgrp group file.txt** changes the group ownership of “file.txt” to the specified group.
umask	Set default file permissions.		• **umask 022** sets the default file permissions to read and write for the owner, and read-only for group and others.

4. Process Management Commands

Commands	Description	Options	Examples
ps	Display running processes.	• **-aux**: Show all processes.	• **ps aux** shows all running processes with detailed information.
top	Monitor system processes in real-time.		• **top** displays a dynamic view of system processes and their resource usage.
kill	Terminate a process.	• **-9**: Forcefully kill a process.	• **kill PID** terminates the process with the specified process ID.

Commands	Description	Options	Examples
pkill	Terminate processes based on their name.		• **pkill process_name** terminates all processes with the specified name.
pgrep	List processes based on their name.		• **pgrep process_name** lists all processes with the specified name.
grep	used to search for specific patterns or regular expressions in text files or streams and display matching lines.	• **-i**: Ignore case distinctions while searching. • **-v**: Invert the match, displaying non-matching lines. • **-r or -R**: Recursively search directories for matching patterns. • **-l**: Print only the names of files containing matches. • **-n**: Display line numbers alongside matching lines. • **-w**: Match whole words only, rather than partial matches. • **-c**: Count the number of matching lines instead of displaying them. • **-e**: Specify multiple patterns to search for.	• **grep -i “hello” file.txt** • **grep -v “error” file.txt** • **grep -r “pattern” directory/** • **grep -l “keyword” file.txt** • **grep -n “pattern” file.txt** In these examples we are extracting our desirec output from filename (file.txt)

7. IO Redirection Commands

Command	Description
cmd < file	Input of cmd is taken from file.
cmd > file	Standard output (stdout) of cmd is redirected to file.
cmd1 <(cmd2)	Output of cmd2 is used as the input file for cmd1.
cmd 1>&2	stdout is redirected to the same place as stderr.
cmd >> file	Appends the stdout of cmd to file.

Arithmetic Operations and Command Line Arguments

Arithmetic Operations	
Command	**Description**
a=10	Assign 10 to variable a
a=`expr $a + 1` or a=$(($a + 1))	Increment the value of a by 1
a=`expr $a \* 5`	Multiply the value of a by 5

Command Line Arguments	
Argument	**Description**
$#	number of command line arguments
$0	name of script
$1 $2 …	First, second, … command line arguments

For Mathematics use following operator in Shell Script

Math- ematical Operator in Shell Script	Meaning	Normal Arithmetical/ Mathematical Statements	But in Shell	
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if expr [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if expr [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if expr [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if expr [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if expr [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if expr [5 -ge 6]

For string Comparisons use

Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Shell also test for file and directory types

Test	Meaning
-s file	Non empty file
-f file	Is File exist or normal file and not a directory
-d dir	Is Directory exist and not a file
-w file	Is writeable file
-r file	Is read-only file
-x file	Is file is executable

Logical Operators
Logical operators are used to combine two or more condition at a time

Operator	Meaning
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR