Linux Commands

Table of Contents

Linux Commands

1) *pwd:* (Print Working Directory)

displays location of current working directory

2) *ls:* (list)

lists all files/directories in current directory

- ls -a → shows the hidden files
- ls -l → Display All Information About Files/Directories
- ls -lt → To sort the file names displayed in the order of last modification time, t means modification time
- ls -ltr → To sort the file names in the last modification time in reverse order, tr means modification time in reverse

3) mkdir <directory_name>: (make directory)

creates new directory

eg) mkdir d1 → creates directory d1
eg) mkdir d3 d4 d5 → creates multiple directories namely d3 d4 d5
eg) mkdir d6/d7/d8 → creates sub directories, d7 is sub-directory of d6 and similarly d8 is sub-directory if d7

4) touch <file_name>:

creates a file without any content

eg) touch f1 f2 f3 f4 → creates empty files namely f1 f2 f3 f4
eg) touch. f10 → creates hidden file

5) cat <file_name>: (concatenate)

displays the content of file in CLI (Command Line Interface) without having to open vi editor.

eg) cat f1 → displays data of f1 in CLI (Linux terminal)

6) *cd:* (change directory)

used to change directory

- cd .. → moving one folder back
- cd ../.. → moving two folders back
- cd - → To go back to the previous directory,
- cd ~/cd → To return to the home directory immediately (~ means tilt)
- cd foldername → go to any directory
- cd ../../Siddesh/→ going 2 folders back and going inside Siddesh folder

7) echo:

used for displaying lines of text in CLI

- echo "this is the devops class" → displays all lines inside " " in terminal
- echo "hi \nhw are you \n this is devops 1st class" → (\n = break line)

- echo -e "hi" "hello" → (-e = input multiple statements)
- echo -e "$1" "$2"

8) vi (or) vim <file_name>: (visual editor)

opens visual editor to add content inside file

- press “i" → (insert mode), modify/ add the content
- exit from the insert mode press "esc"
- :wq! → (write & quit), save
- :q! → (quit), not save
- ESC + dd → delete the complete line in which our cursor is there
- :set nu or :set number → Show Line Numbers in Vi editor
- :set nonu → To disable the line numbers off
- :/pattern → to find any word in vi editor
  eg) :/Testing
  Press "n" to go to next same pattern present in file
- :%s/Git/Git_Hub/g → replace Git with Git_Hub, % means all, s means substitute, g means global
- :1s/Git/Git_Hub/g → replace pattern only in 1st line
- :1,3s/Git/Git_Hub/g → replace pattern in 1st to 3rd line
- :10,20s/testing/development/g → replace pattern in 10th to 20th line
- :20,$s/linux/unix/g → replace pattern in 20th to last line, $ means till last line
- :1s/Git/Git_Hub/1 → replace pattern in 1st line and 1st occurrence only, giving no at last like 1,2, etc... is occurrence

9) wc <file_name>: (word count)

count the number of characters, words, lines, and bytes in a file

- wc -l f1 → (-l = lines), lists total no of lines
- wc -c f1→ (-c = characters), lists total no of characters
- wc -w f1→ (-w = words), lists total no of words

10) append: (>>)

adds/apends content to the end of a file without erasing existing data.

- ls -lart >> file2 → (-lart = lists hidden files, to sort the file names in the last modification time in reverse order), all files displayed will be appended at bottom of file2 without overwriting existing data.
- echo "hi \nhow are you" >> file2 → echo “statement” will be added in file2 at end/bottom

11) redirect: (>)

Will be redirected to that file, creates and copied to new file but if file is present then it will overwrite in that file.

eg) ls > f1 → list of files in ls will be copied to file1

[**Note:** If file is present entire data will be overwritten, if file is not present then it will create new file and save in it]

12) *du:* (disk usage)

shows memory used by the directory

- du -sh Downloads → (sh = display the total disk usage of a directory)
- du -sh * → shows size of all files in that directory

13) *df:* (disk free)

displays information about total memory used and available memory on a file system

- df -h → shows total memory used and memory available (entire system)

14) free:

used to check RAM size. The free command gives information about used and unused memory usage and swap memory of a system.

By default, it displays memory in kb (kilobytes)

- free -m → display information in Megabytes (MB)
- free -g → display information in Gigabytes (GB)

15) head and tail:

the head command will output the first part of the file, while the tail command will print the last part of the file.

- head f1 → shows top 10 lines of file by default
- head -5 f1 → to show first 5 lines of file
- head -1 f1 → shows first line of file

- tail f1 → shows bottom 10 lines of file by default
- tail -15 f1 → shows bottom 15 lines of file
- tail -1 f1 → shows last line of file

16) pipe: (|)

The pipe takes output from one command and uses it as input for another.

- ls -lart | wc -l → (| = pipe, wc = word count, l = lines), displays no of lines in ls

[Note: To display any specific line like 20 then we can use pipe]

- head -20 f1 | tail -1 → displays 20th line
- head -20 f1 | tail -1 | wc -l → (l = lines), displays no of lines in 20th line
- head -20 f1 | tail -1 | wc -c → (c = characters), displays no of characters in 20th line

- head -20 f1 | tail -1 | wc -w → (w = words), displays no of words in 20th line
- ls -lart | wc -lcw → display lines, characters and words of ls

17) sort <file_name>:

to sort the output of a file in given order, by default sort is in ascending order.

- sort f1 → sorts data in ascending order
- sort -r f1 → (r = reverse), data get sorted in descending order

18) *grep:* (global regular expression print)

search for matching patterns in a file and displays entire line where pattern is present (used only for pattern search, doesn't replace)

➢ -i, -n, -c, -w, -l:

- -i → case insensitive
- -n → print line no like 1,2, etc...
- -c → count of pattern
- -w → search specific word like devops and excludes if that word is present in any other words like awsdevops
- -l → displays files and directories where pattern is present
- grep -i "devops" file2 → displays all lines where pattern is present and is case insensitive so can use both small and big letters for pattern search
- grep -ic "Devops" file2 → displays count of patterns(devops) present in file2
- grep -in "DEVOPS" file2 → prints line numbers
- grep -w "devops" file2 → displays specific words
- grep -il "devops" * → (* = all), displays all files and directories where pattern is present
- grep -ilR (or -i -l -R) → (R = Recursive), searches pattern even in sub-directory
- grep -e "dev" -e "ops" → -e is used to search multiple patterns
- grep ^h f2 → (^ = search only first letter/word in a line), displays lines that match with pattern (i.e, h) at first
- grep ^hi f2 → displays lines that match with hi
- grep s$ f2 → ($ = search only last letter/word in a line), displays lines that match with pattern (i.e., s) at last

19) uniq:

deletes duplicate/repeated lines in a file

- sort file | uniq → removes duplicate files and shows only unique file
- sort file | wc -l → displays lines in file/file_name
- sort file | uniq | wc -l → displays lines in file removing duplicates

- sort file | uniq > file1 → (> = redirect), saving uniq command file data to another file name file1
  (Here, original file is file and new file is file1 where no duplicates are present…. Won't modify or overwrite original file instead create new file)

[**Note**: Since, uniq command is not present we need to install it so as to use it.
It's used in ubuntu user to remove duplicate files (AWS console linked)
sudo apt install uniq
sudo = used in ubuntu as admin/root, apt = keyword to install in ubuntu]

20) *chmod:* (change mode)

used to provide permissions to a file like read, write and execute.

permissions	owner			Group			Others		
Binary no	Read (r)	Write (w)	Execution (x)	r	w	x	r	w	x
	4	2	1	4	2	1	4	2	1

- chmod 760 file → ▪ (760 = owner (7 = read (4), write (2) & execute (1) so access to read, write and execute)
  ▪ group (6 = read (4), write (2) so access to read and write only)
  ▪ others (0 = no access) for file\file_name
- chmod 544 file → ▪ giving permission for file/file_name as follows:
  ▪ (owner: 5 = read & execute access, group: 4 = only read access, others: 4 = only read access)
- chmod 640 file file1 → giving permissions to multiple files in single command
- chmod 766 test/ → giving permissions to directory (owner: all, group: read & write, others: no access)
- chmod -R 777 test/ → (R = recurrsive, applies to all sub folders/files in directory), giving permission to directory and its sub folders/files
- chmod 777 test/f1 → giving permission to only file, f1 which is inside test directory

➢ **Different approach to give permission *(owner = u, group = g, others = o)***

- chmod g+x,o+x f3 → for f3 file included execution permission for group (g+x) and execution permission for others (o+x)
- chmod g-x,o-x f3 → removed execution permission for group (g-x) and others (o-x)
- chmod a-x f3 → for all 3(owner(u), group(g)&others(o)) at once removing permission for execution for f3
- chmod g-w,o-w f3 → remove permission to write for group & others for f3
- chmod a+w f3 → add permission to write for all (u, g, o) for f3
- chmod o-rw f3 → remove permission to read and write for others for f3
- chmod u+rwx f3 → add permission to read, write & execute for owner for f3 at once

[Note: for owner use u, don't touch permission of owner until requirement is there]

21) *chown:* (change owner)

change ownership of file/directory

- sudo chown root:root folder2/ → changing ownership for both for owner and group as root for folder2
- sudo chown root:root f2 → changing owner for both to root in f2
- sudo chown ubuntu f2 → changing only for owner in f2
- sudo chown root:ubuntu f2 → changing root for owner and ubuntu for group in f2

22) umask: (user file-creation mode mask)

set the default permission (i.e, read, write & execute)

- 777-002 == 775 (002, owner = full permission, group = full permission, others = read and execute)
- 777-114 == 663 (114, owner = rw, group = rw, others = wx)
- 777 - 001 == 776 (001, owner = rwx, group = rwx, others = rw)
- umask u(rw) =, g(rx)=, o(r) = 123 (chmod = 654)
- umask -S --> "-S" is used to check what permission is given to umask to make default permission
- umask 000 (owner(u) = rwx, group(g) = rwx, others(o) = rwx)
- umask 001 (u = rwx, g = rwx, o = rw)
- umask 114 (u = rw, g = rw, o = wx)

23) tee:

similar to pipe, used to read from standard input and write to standard output in files simultaneously

- ls -lrt | tee log1 → displays/reads output first and then its output is overwritten in file name log1
- ls -lrt | tee -a log1 → (a = append) displays output and appends output to file log1

[Note: using only tee will redirect(overwrites), using -a in tee will append(add)]

1. **Steps to create instance in AWS console in EC2,**
   (Select EC2 —> Launch instance —>
   Name —> any name like linux practice
   Application and OS —> Ubuntu
   Key Pair—> create new key pair —> name —> create key pair
   Click Launch instance)

2. **Linking AWS instance in GitBash to work in ubuntu**
   (ssh -i pemfilename.pem ubuntu@publicIP)

24) *rm:* (remove)

used to remove files and directories

- rm f1 → remove file
- rm -r d1 (or) rmdir d1→ remove directory

25) *mv:* (move)

used to rename files/directories and also to move any files/directories (cut and paste)
If directory is present then it will copy, if not then will rename.
(**Syntax:** mv source renamefile)

- mv f2 file2 → renames f2 as file2 (renaming file)
- mv test2 folder2 → renames test2 as folder2 (renaming directory)
- mv f3 d3 → moving file to directory
- mv d1 d3 → moving directory to directory

26) *cp:* (copy)

used to copy file to any folders/directories
(**Syntax:** cp source destination)

- cp f3 test1 → copy f3 file to test1 directory (copy file)
- cp f1 file file1 test → copy multiple files (f1 file file1) to test directory
- cp ../../file . → Copy file that is 2 paths back to current directory
- cp /home/ubuntu/file1 . → copying file1 which is in home directory(/home/ubuntu/file1) to current directory(.)
- cp /home/ubuntu/file1 /home/ubuntu/test → copying file1 which is in home to test directory
- cp -r test test1 → (-r = recursive copy), used to copy entire directory to another directory (copy directory)
- cp -p f1 test → (-p = permissions), it will copy with same permissions
- cp -rp test test2 → combination of recursive copy with same permissions
- cp d1/d2/* d1/ → (* = all), copying all files in d2 which is sub-directory of d1 to d1
- cp -i f2 f3 → (-i used to ask for overwrite or not in copy command), without using -i, file will be overwritten by default
- cp -p f2 f9 → (p = permission, to copy permission of file), copies f2 permission to f9

27) diff <file1> <file2>:

compare two files line by line

eg) diff f77 f2 → show only difference in the files, won't show same content

28) sed: (Stream Editor)

perform lots of functions on file like searching, find and replace, insertion or deletion of text in original file (without opening VI editor)

- sed 's/devops/shs/g' f77 → (s = substitution, g = global), replacing devops word with shs in f77 file (original file not affected)
- sed '2s/devops/shs/g' f77 → replacing devops word with shs in f77 file but only in 2nd line
- sed 's/devops/shs/i' f77 → (i = case-insensitive in sed command), can use ig too where words get replaced globally (throughout file)
- sed '2s/devops/shs/ig' f77 → ig makes search case-insensitive and devops is replaced by shs in 2nd line only
- sed -i 's/devops/shs/ig' f77 → (-i = insert, saves in original file but won't display), replaces devops to shs in f77 in orginal file without opening VI terminal
- sed '2,4s/devops/shs/ig' f77 → replaces devops to shs in f77 but only between line 2-4
- sed '4,$s/devops/shs/ig' f77 → ($ = end of file/till last line), replaces devops to shs in f77 but only between line 4th to last line
- sed -n '5p' f77 → (n = used to display particular line only), displays 5th line only
- sed -n '4,6p' f77 → displays 4th and 6th line only
- sed -n '4,$p' f77 → ($ = end of file/till last line), displays from 4th line to last line
- sed -n '3p' f77 ; sed -n '6p' f77 → displays only 3rd and 5th line
- sed '3d' f77 → (d = delete), deletes 3rd line from f77 file
- sed '3,5d' f77 → deletes 3rd to 5th line
- sed '3,$d' f77 → deletes 3rd to last line

29) cut:

used for cutting out the sections/columns of a file from each line of files and writing the result to standard output.

- cut -d " " -f1 file2 → (d = delimiter, " " = field separator, need to give field separator like space, colon, etc., f1 = col no), shows 1st col in f1 separated by space (field separator)
- cut -d " " -f2 file2 → cuts and displays 2nd col
- cut -d " " -f3 file2 → cuts and displays 3rd col
- cut -d " " -f 1-3 file2 → cuts and displays 1st to 3rd col
- cut -d ":" -f1 file2 → cuts and displays 1st col separated by :(field separator)

30) awk:

Awk is mostly used for column search and processing.

It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

To overcome disadvantages of cut, awk is used (in cut we can't cut col from last)

(**syntax:** awk -F "field separator" '{print$col_no}' file_name**)**

• awk -F " " '{print $1}' cut	→	displays 1st column
• awk -F " " '{print $2}' cut	→	displays 2nd column
• awk -F " " '{print $3}' cut	→	displays 3rd column
• awk -F " " '{print $NF}' cut	→	displays last column
• awk -F " " '{print $(NF-1)}' cut	→	displays last-1 column
• awk -F " " '{print $1,$3}' cut	→	displays 1st and 3rd column having "space" as field separator

(Note: if there isn't any field separator in any row then whole row will be displayed)

31) top:

displays real-time information about processes that are currently running.

Can check lot of details like no of users, load average, total no of processes, CPU utilization, etc.

(ctrl + c --> to come out from loop(exit) as it keeps on refreshing continuously)

32) uptime:

the command returns 4 key pieces of information:

- the current time,
- how long the system has been running,
- no of users currently logged in
- the system load averages

(a) diff b/w cpu utilization and load avg:

load average shows how many processes are waiting for a cpu core while %CPU indicates how busy the cores are.

cpu utilization shows amount of work handled by CPU

(b) load avg 0.0, 0.1, 0.0:

- shows the number of tasks currently executed by the CPU and tasks waiting in the queue. (or)
- Load Average gives you a historical view of how busy your system has been. A higher load average means that more processes have been waiting to use the CPU.
  - The first value depicts the last-minute average load.
  - The second gives us the last 5-minute average load.
  - The third value gives us the 15-minute average load.

(c) diff PID & PPID:

PID stands for Process ID, which means Identification Number for currently running process in Memory.

PPID stands for Parent Process ID, which means Parent Process is the responsible for creating the current process (Child Process).

(d) diff orphan process zombie process:

An orphan process is formed when its parent dies while the process continues to execute, while a zombie process is a process that has terminated but its entry is there in the system. (or)

A zombie process refers to a process that has completed its execution and waiting for its parent to collect its exit status, whereas an orphan process is still in its execution phase even after its parent terminates.

33) ps: (process status)

- ps -ef → list of all processes running on the system
  - ps -ef|grep -i "java" → to filter java process
- ps -u "username" → to check processes started by any user
  - ps -u "ubuntu" → displays psrocesses started by ubuntu user

Install apache2 app
sudo apt update
sudo apt install apache2 (install apache)

34) kill:

used to terminate running processes.(forceful killing)

kill -9 PID → to forcefully stop process using process ID(PID)

eg) kill -9 1576 → forcefully stops 1576 process

[Note: never use kill until get requirement from high authority]

35) service:

to manage system services

- sudo service service_name stop → to stop process gracefully using service
  - sudo service apache2 stop
- sudo service service_name start → to start process gracefully
  - sudo service tomcat start
- sudo service service_name restart → to restart process gracefully
  - sudo service apache2 restart
- sudo service service_name status → to check status(active/inactive) of process
  - sudo service apache2 status

36) tree:

to display the directory structure in a tree-like format

(shows all directories, sub directories, files and sub-files in tree structure)

[**Note:** since, tree command is not available so need to install
sudo apt update
sudo apt install tree]

37) links:

(a) softlink or symbolic link or sym link:

creating reference/shortcut to existing file.

if we make any changes in original file, it will get reflected in softlink. In case if we delete original file, then softlink won't work.

(Both the hard link and the original file have its own i-node)

eg) ln -s /home/file link1 → (s = softlink), file is created as softlink (link1)

(b) hardlink:

it's just a reference/shortcut to a file like softlink.

if you make any changes in original file, it will get reflected in hardlink. If we delete original file, hardlink will work because it points to i node of a file.

(Both the hard link and the original file share the same inode)

'i-node'

ls -i filename → to get i node of a file

eg) ln /home/file link2 → file is created as hardlink (link2)

[**Note:** In general, hardlinks are not created in most of the cases, very rarely it's created]

38) find:

used to find location of the file, by default it is recursive (folder and sub folder) search

- find . -iname "test" → (. = present working directory or can give /home so it will search from home), displays present working directory to file present inside directory or sub directory
- find . -type f -mtime +10 → f = files, +10 = check files that was created 10 days before
- find . -type f -mtime -10 → f = files, +10 = check files that was created within 10 days
- find . -type d -mtime -10 → similarly for directory, d = directory, +10 = check directories created 10 days before and -10 = check directories created within 10 days
- find . -type f -mmin +15 → mmin = minutes, +15 = any files created 15min before
- find . -type f -mmin -15 → mmin = minutes, -15 = any files created within 15min
- find . -type f -perm 0777 → perm = permission, any files having 0777 permission will be listed
- find . -type d -perm 0777 → perm = permission, any directories having 0777 permission will be listed

- find . -perm 0666 → gives both files and directories having 0666 permission if files/directories aren't mentioned
- find . -type f -empty → displays empty files
- find . -type f -not -empty → displays non-empty files
- find . -type d -empty → displays empty directories
- find . -type f -mtime -10 -not -empty → files created within 10 days and not-empty files
- find . -maxdepth 1 -iname "test" → (1 = search 1 directory, current directory), used for limiting search to a specific directory
- find . -maxdepth 2 -iname "t1" → check in 2 directories, current and sub-directory

[**Note:** If we don't use maxdepth then by default it will do recursive search, maxdepth is used to set the no. for searching directories]

39) xargs: (extended arguments)

converts input from standard input (STDIN) into arguments to a command.

eg) find . -type f -mtime +300|xargs rm → deletes all files that were created before 300days

40) uname -a:

linux distribution version, private IP

41) who:

shows no. of users logged into the system showing user, date, time and IP

(If two tabs are opened then it's 2 users)

- who |wc -l → to list no of counts of users
- whoami → check users like ubuntu, etc

42) ssh: (secure shell)

tool for remote administration and file transfer, used to connect with remote server.

- ssh [secure shell]: port 22 → port22 as its default port of ssh
- ssh user@serverIP → connect with server
- ssh ubuntu@192.13.14.15
- ssh -i test.pem ubuntu@12.21.11.1 → connect with server

43) 3 types of authentications:

a) pem file
b) Password
c) Password less

b) steps for password authentication:

Go to any remote server (ubuntu@serverIP)

1.Create new user (in ubuntu user)

(sudo adduser dev)

(cat /etc/passwd) → Validate user

(su dev) → (su = switch user), switch the user

2.add that user to sudo group (in ubuntu user)

(exit) → so as to add user to sudo group

(sudo adduser dev sudo) → adding user to sudo group, this syntax works only for ubuntu system

(cat /etc/group) → validate user added to group

(su dev) → switch to dev user

3.sudo vi /etc/ssh/sshd_config (in ubuntu user)

Comment the below line

Include /etc/ssh/sshd_config.d/*.conf

#Include /etc/ssh/sshd_config.d/*.conf (add hash)

#PasswordAuthentication yes

PasswordAuthentication yes (remove hash)

#PermitRootLogin prohibit-password

PermitRootLogin yes (remove hash)

4. restart the ssh service

sudo service ssh restart

5. connect to the user using password authentication

ssh test@Public_ip_adress

Password: ****

c) Passwordless Authentication:

Method1

Passwordless auth between local to remote:

1. generate the key on local system (local user, system)

ssh-keygen

(or)

cd .ssh/ --> go to .ssh/ path

ls --> shows keys that's stored in .ssh/

2. copy the public key from local to remote (local user, system)

cd .ssh/

ls

cat id_rsa.pub (copy key)

3. login to remote server then paste the public key content (remote server, yatish)

cd .ssh/

vim authorized_keys (paste key)

exit

4. Now try login without password

ssh user@ip_address

Method2

(Passwordless Auntentication)

1. generate the key on local system/remote system
   ssh-keygen
2. go to .ssh path
   cd .ssh/ → (remote user)
   (To find .ssh file:
   cd
   ls -a
   cd .ssh/) → (local user)
3. copy public key with its path to user that needs passwordless authentication
   ssh-copy-id -i /c/Users/yatis/.ssh/id_ed25519.pub yatish@13.232.104.36
4. give password for 1st time only of that user
   eg) User: yatish@serverIP
   Password: yatish123
5. login to user
   ssh username@serverIP

44) *scp:* (server copy)

used to copy files/directories from one server to another server
(local to remote and remote to remote)

Syntax: scp -i test.pem local_file ubuntu@server:path/

- scp -i file user@server:/home/ubuntu → (-i = identify file), copying file from local to remote server
  eg) scp -i linuxcommand.pem f4 ubuntu@35.154.179.142:/home/ubuntu)
  - password: ***
- scp -i -r dir1 user@server:/path → copying directory from local to remote server
  eg) scp -i linuxcommand.pem -r d4 ubuntu@35.154.179.142:/home/ubuntu)
  - password: ***
- scp -p file1 user@serverIP → (-p = permission), copy permission of file1 from local to remote which is having same file

45) *rsync:* (remote sync)

Similar to scp, used to copy files/directories within the server and also between the servers (server to servers). It will start from where it got stopped if copying fails for any reason then it will compare and copy.

Syntax: rsync -avz --progress -e "ssh -i test.pem" files ubuntu@server:/path →
(a = copy data recursively, v =process data, z = compress the data, --progress = to check progress), can just use -a instead of -avz

(eg) rsync -avz --progress -e "ssh -i linuxcommand.pem" -r d3 yatish@35.154.179.142:/home/yatish

Diff between scp and rsync:

- while copying the data from one server to another server, if copy is stopped due to network issue/system issue. if we reinitiate copy using scp the it will start coping from the beginning.
- Instead of scp if I use rsync, it will start copying from where it was stopped because it will compare and copy.

46) nohup [no hang up]:

used to ensure a process continues running even after the terminal session that launched it is closed or exits.

(useful for long-running tasks or background processes that you don't want to be interrupted by accidental terminal termination)

Syntax: nohup ./script.sh &

47) ping ip_addr or ping dns_name:

check status of the network/status

48) netstat -na:

shows system ports details

49) netstat- na |grep 8080:

to check specific port status

50) sudo hostnamectl set-hostname My-TestBox:

to change hostname to required name (instead of IP address, host name will be shown)