LINUX COMMANDS

VRX

Difference between Shell and terminal:

Shell

The shell is a command-line interpreter that takes user commands and executes them. It acts as a bridge between the user and the operating system, allowing you to interact with the OS by typing commands.

Terminal: Terminal is a tool where u can pass your shell commands. A terminal, or terminal emulator, is a program that opens a window where you can interact with the shell. It's essentially the interface that displays text input and output.

Linux Commands:

1. Ls:

The `ls` command in Bash is used to list the contents of a directory. It displays the files and directories within the specified directory or, if no directory is specified, within the current working directory.

Basic Usage

ls [OPTION]... [FILE]...

- **No Options:** List the contents of the current directory.
  Ls
- **-l** (Long Listing Format): Shows detailed information about each file, including permissions, number of links, owner, group, size, and timestamp
  Ls -l
- **-a (All Files):** Includes hidden files (those starting with a dot .)
  Ls -a
- **-h (Human-Readable):** Used with -l to display file sizes in a more readable format (KB, MB, GB)
  Ls -lh
- **-R (Recursive):** Lists files in all subdirectories recursively.
  Ls -R
- **-t (Sort by Modification Time):** Sorts the files by modification time, most recently modified first.
  Ls -lt
- **-r (Reverse Order):** Reverses the order of the sort.
  Ls -lr
- **ls -alh** would list all files (including hidden ones) with detailed information in a human-readable format.

2. cd:

The `cd` command in a shell stands for "change directory." It's used to navigate between directories (folders) in a command-line environment. Here's how it works:

1. Change to a Specific Directory:

- To move to a specific directory, you use `cd` followed by the path to that directory.

```
cd /path/to/directory
```

For example, to navigate to the `/home/vipul` directory:

```
cd /home/vipul
```

2. Change to the Home Directory:

- If you type `cd` without any arguments, it will take you to your home directory.

```
cd
```

- Alternatively, you can use `cd ~` to achieve the same result.

3. Move Up One Directory Level:

- Use `cd ..` to move up one level in the directory tree (i.e., to the parent directory).

```
cd ..
```

- For example, if you're in `/home/vipul/projects`, `cd ..` will take you to `/home/vipul`.

4. Navigate Using Relative Paths:

- You can move to a directory relative to your current location. For example, if you're in `/home/vipul` and want to go to `projects`, you can just type:

```
cd projects
```

- This assumes that the `projects` directory exists within `/home/vipul`.

5. Change to the Previous Directory:

- To go back to the previous directory you were in, use `cd -`.

```
cd -
```

- This will switch between your current directory and the last one you were in.

Examples

- Change to `/var/log` directory:

```
cd /var/log
```

cd ../ .. to move two levels up

3. cat

Displaying text files, displaying combed text files.

- **Cat + enter “word”:** it is used to echo text. To exit press ctrl +d.
- **Cat file1 file 2:** it will display the content of files.
- **Cat -b file1 file2:** it will display content of line with line number.
- **Cat -n file1:** it will add line number to the gap line too(line space).
- **Cat -s file1:** it will treat multiple line gaps as single gap in your file.
- **Cat -E file1**: to add the $ symbol to end of each line to let us know where line ended.

I/O Redirection:

To transfer a output into a file.

- **Cat > filename:** when u will run the command it lets u write inside the file without going inside a text editor. I if run the same command again it will replace the earlier text with new text provided.
- **Cat >> filename:** this command will add text to the file without overriding it.
- **Cat file1 file2 > new file:** this is combining the content of both files and copy it in 3rd file. Abasically to copy one file to another file etc.
- **Cat file1 >> file2:** it will copy content of file 1 to file 2.
- **Ls -al > output.txt:** will write the output of ls command in output.file. you can use this with any command of Linux.

4. mkdir:

To make folder/directories in Linux.

- **mkdir dir name:** to create a directory at current location you are.
- **mkdir path/dir name:** to create a director at given path.
- **mkdir -p path/dir_name:** This will create path which doesn’t exists before and create a directory inside that path.
- **mkdir -p path/{dir1, dir2,dir3}:** This will help u to create multiple directories at given path. (don’t enter space after commas).

5. rm and rmdir:

to remove directory or file.

- **rmdir dirname:** to delete a directory
- **rmdir pathtodir:** to delete a dir. located at that path.
- **Ls -R** will show you the directory structure.
- **rmdir -p pathtodir:** will delete the whole path along with directory.

- **rmdir -pv pathtodir:** v is for verbose; it gives information about what's happening in the background.
- **rm -r:** same same.
- **rm -rv:** to delete a file present in that directory and directory too.

6. Cp:

- **cp** [option source destination].
- **cp filel file 2**: copy filel to file2
- **cp file direc/:** to copy file into the directory.
- **cp filel file 2 directory**: to copy multiple files into directory.
  when u have file present with same name and u copy again it will overwrite it automatically.
  in order to get a permission to ask for overwrite we use a flag

- **cp-1 filel dir/:** to get ask for overwrite of file if file is already present there
  cp../file.txt../file2.txt dir/: to copy files from parent directory to desired directory
- **cp-R dirl dir2:** to copy content of dir1 into dir2.
- **cp-VR:** verbose.

7. mv:

to move files

- **mv filel file2:** to mv content of 1 file to other content
- **mv file dir/:** to move file into the directory
- **mv-i filel dir/:** to get interactive mode
- **mv diri/ dir2/:** to mv dir to another dir
- **mv -v file dir/:** to get detail of what's happening in background.

8. Less:

It is used to view the files with large text.

Less filename: It will display the content of file and You can use arrow key to see the content.

- When you press Space then it will display the content page wise (Downwards).
- When you press b then it will shift to next page (upwards) 1 page at a time.
- When you press G it will take you to end of file.
- When you press 1G or g it will take you to top.
- To search for a word in the file we use "/word" then use "n" to go the next word (from up to down).

- To search for a word in the file we use “?word” then use “n” to go the previous word (Down to up).
- Q to quit the file.

9. Touch:

To create a new empty file.

Touch filename: to create a empty file. You cannot create directory by using touch.

Touch filename. Extension: to create a file with the desired extension.

Touch filename (already created file): to change the timestamp of the file.

10. SUDO:

Sudo command which rules other command.

Sudo means Super User Do.

It gives extra privileges.

11. Top:

Provide a real time view of what process are being running and which process is taking how much cpu and its refreshes every 3 seconds.

It has:

PID: Process ID.

COMMAND: Then process which is running.

CPU: CPU usage.

In order to change the refreshing rate, press S and the rate and press enter.

When you press i it will show the running process avoiding the ideal process.

When you press k and give the process id and press enter to kill a process.

12. Kill:

To kill the process.

Kill [flag] PID

Pidof process name: this is to get paid of process.

Kill pid: to kill process.

Kill -KILL pid: this will force the process to close.

Kill -9 pid: kill the process which are not being killed by normal kill process.

ps -ux : this will give u list of all running process.

ps -U username: to get the process run by the user provided.

Ps -C process name: to list out instances related to the process provided.

13. Echo:

It is used in bash scripting.

echo "Word" : to echo a word.

echo -e 'some \text": some   ext.

echo -e 'some \n text': some

14. File Permission Symbolic permission and chmod:

-rw-rw-r—1 vipul Vipul 74 jun 11 13:07 abc.txt

Here,

First dash (-) represent the type of file.

rw- represent the permission of owner.

rw- represent the permission of provided for group.

r—represent the permission for all other user.

1 symbolic link present.

Vipul owner

Vipul group

74 size of file.

Jun 11 : date

13:07 time

Abc.txt: file name.

File security and permission:

A file is owner by user who create it

That user can then specify who can read, write and execute that file. A file when created can be accessed in three ways: Reading, Writing, Executing.

Chmod [Who] operator [permission] filename:

Who means:

U the user permission.

G the group permission.

O the other permission.

A means all (user, group and other)

Operator means:

+ add a permission

(-) Take away permission

= set permission

Permission means:

R read permission

W write permission

X execute permission.

Chmod u=rw, g+w, 0-w file name: to provide different permission to different users.

Directory permissions and chmod:

Read: Grants the capability to read i.e. to view the content of file.

Write: Grants capability to modify, or remove the content of the file.

Execute: User with execute permission can run a file as a program.

Octal and Numerical Permission (chmod):

The chmod numerical format accepts up to four octal digits.

The three rightmost digits refer to permission for file owner, the group, and other users.

The optional leading digit (when 4 digits are given) specifies the special setuid, setgid and sticky flags

Permission	rwx
7 read, write and execute	rwx
6 read and write	rw-
5 read and execute	r-x
4 read only	r--
3 write and execute	-wx
2 write only	w
1 execute only	--x
O none	---

Chmod +755 file name:

7 is for owner

5 is for user

5 is for other

15. Which and whatis:

Whatis – display one-line manual page descriptions

Each manual page has a short description available within it

whatis search the manual page name and displays the manual page descriptions of any name matched.

Which – shows the full path of (shell).

16. Useradd:

useradd - create a new user or update default new user information

user add [options] LOGIN

user add -D

user add -D [options]

useradd is a low-level utility for adding users. On Debian, administrators should usually use **adduser**(8) instead. When invoked without the -D option, the **useradd** command creates a new user account using the values specified on the command line plus the default values from the system. Depending on command line options, the **useradd** command will update system files and may also create the new user's home directory and copy initial files. By default, a group will also be created for the new user (see -g, -N, -U, and USERGROUPS_ENAB).

useradd mark -m -s /bin/bash -g [users] -c “comments”

we need sudo access to create a user.

Passwd: to change the password for root user

Sudo Passwd username: it will allow to change password for given user

17. Userdel:

userdel - delete a user account and related files

userdel [options] LOGIN

DESCRIPTION userdel is a low-level utility for removing users. On Debian, administrators should usually use deluser instead. The userdel command modifies the system account files, deleting all entries that refer to the user’s name LOGIN. The named user must exist.

When you remove a user using above command only user is removed but his work his home directory is present. In order to remove the its work too use

Sudo userdel -r user:

18. Basics of Group Management (groups):

groups - print the groups a user is in

groupadd - create a new group

groupdel - delete a group

gpasswd - administer /etc/group and /etc/gshadow

The options which apply to the gpasswd command are:

-a, --add user

Add the user to the named group.

-d, --delete user

Remove the user from the named group.

Sudo gpasswd -a group user name group name: to add a particular to a group.

.bashrc file:

The .bashrc is a configuration file for the Bash , which every time you open a terminal is also loaded. In it you can abbreviations for commands (ie an alias) Enter, you change here is the appearance of the prompt , or you can add additional scripts which facilitate a working with the Shell.

The bash profile can change as well and fill with the same content as the bashrc, so this article almost 1 may: 1 for bash profile be taken. The difference is that. bashrc runs each time you start an interactive shell as a terminal and the bash profile only when you start a login shell.

If one wants to perform something once when a login shell is the bash_profile right place. Will you, that changes every time you start a terminal or a console be executed is the bashrc right place.

19. Viewing Resources (du, df, free command):

du - estimate file space usage

du [OPTION]... [FILE]...

df - report file system disk space usage

df [OPTION]... [FILE]...

-h, --human-readable

print sizes in human readable format (e.g., 1K 234M 2G)

free - Display amount of free and used memory in the system

du -h: to display file usage in human readable format.

du -sh: to get total memory used.

df -h

df -sh

du -sh /folder name:

Free -k -m -g:

-k -m -g = for bytes

20. Watch command:

watch - execute a program periodically, showing output Fullscreen

watch [options] command

watch runs command repeatedly, displaying its output and errors (the first screenful). This allows you to watch the program output change over time. By default, the program is run every 2 seconds. By default, watch will run until interrupted.

Use ctrl + c to come out of that command

You can use this with any other command

Watch free -m

Watch ls -al.

To change the interval we use -n flag

Watch -n 1 ls -al : it will list after 1 sec

21. Head & Tail commands:

head - output the first part of files

usage:

head [OPTION]... [FILE]...

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

tail - output the last part of files

usage:

tail [OPTION]... [FILE]...

Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

Head filename:

Tail filename:

To print the number of lines

Head -n3 file name:

Tail -n3 filename:

Tail -f filename: it shows u last 10 lines and will keep showing the last lines even after changing files.

Head file1 file2 file3

Head -3 file1 file2 file3.

22. Find:

find - search for files in a directory hierarchy

find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]

This manual page documents the GNU version of find. GNU find searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left

hand side is false for and operations, true for or), at which point find moves on to the next file name.

Find /folder path -name file name:

Find /home/vipul -name test.sh

Find /home/vipul -name test.*

Find /home/vipul -name *.sh

Find /home/vipul -name test*:

Find / -name filename:

Find /foldername -mtime 1: this will find all the files created 1 day before.

23. WC Command:

wc - print newline, word, and byte counts for each file

wc [OPTION]... [FILE]...

wc [OPTION]... --files0-from=F

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. With no FILE, or when FILE is -, read standard input. A word is a non-zero-length sequence of characters delimited by white space. The options below may be used to select which counts are printed, always in the following order: newline, word, character, byte, maximum line length.

-c, --bytes

print the byte counts

-m, --chars

print the character counts

-l, --lines

print the newline counts

--files0-from=F

read input from the files specified by NUL-terminated names in file F; If F is - then read names from standard input

-L, --max-line-length

print the length of the longest lines

-w, --words

print the word counts

--help display this help and exit

--version

output version information and exit

wc file name:

24. Cal commands:

cal, ncal — displays a calendar and the date of Easter

cal 2016: will display 2016 calendar

cal 2 2014: will display the feb month of 2014.

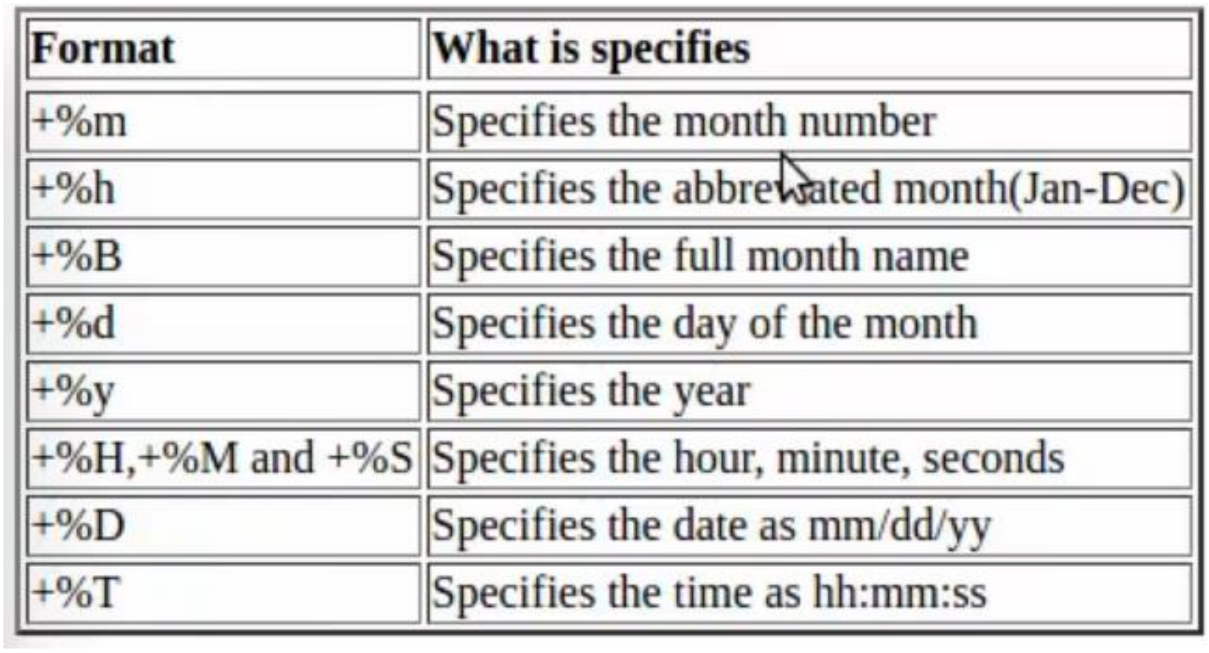
Cal -3 : will display the the current month previous month and next month.

25. Date:

date - print or set the system date and time

date

date -s "11/12/2001 12:56:00": will set the system date.



Format	What is specifies
+%m	Specifies the month number
+%h	Specifies the abbreviated month(Jan-Dec)
+%B	Specifies the full month name
+%d	Specifies the day of the month
+%y	Specifies the year
+%H,+%M and +%S	Specifies the hour, minute, seconds
+%D	Specifies the date as mm/dd/yy
+%T	Specifies the time as hh:mm:ss

Date +%m

26. Running multiple commands in Terminal:

How to Run Two or More Terminal Commands at Once in Linux

Option One: The Semicolon (;) Operator

ls ; pwd ; whoami

Option Two: The Logical AND Operator (&&)

ls && pwd && whoami

Option Three: The Logical OR Operator (||)

ls || pwd || whoami

Difference between ; and &&

; : it runs every command given regardless of success or failure off previous command.

&&: it will not execute the next command if previous commands fail.

||: it will run the command provided if first fails then next command will be runned I first runs successfully then next command will not be executed.

27. Apt-get command:

apt-get - APT package handling utility -- command-line interface.

28. ifconfig command:

ifconfig - configure a network interface

ifconfig [-v] [-a] [-s] [interface]

ifconfig [-v] interface [aftype] options | address ...

Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, ifconfig displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

Address Families

If the first argument after the interface name is recognized as the name of a supported address family, that address family is used for decoding and displaying all protocol addresses. Currently supported address families include inet (TCP/IP, default), inet6 (IPv6), ax25 (AMPR Packet Radio), ddp (AppleTalk Phase 2), ipx (Novell IPX) and netrom (AMPR Packet radio)

29. tar Command:

tar -cvf tarname foldername: -c is for creating

tar -xvf tarfile : -x is for extracting.

tar -czvf test.tar.gz test: to create a gzip file.

tar -xzvf test.tar.gz

30. Grep:

Global regular expression

It processes text line by line and return if found.

grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

grep "options" file.txt

Here options is the word to find in file file.txt.

grep -I "options" file.txt: It will give both lower case and upper case.

grep -n “option” file.txt: it will print the line number

You can use multiple files too.

grep -v “some Options” file.txt: it will show the lines which doesn’t contains the “Some Options” word.

31. Netstat: Network Statistics

Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

Netstat prints information about the Linux networking subsystem

netstat -a: it shows all the connections which are available on system including tcp, udp also status

netstat -a | less: better view

netstat -at |less: shows tcp connections (-t) (-u) for udp connections.

netstat -l | less: to show ports which are listening (-t) (-u) for udp connections.

netstat -s | less: to show statistics (-t) (-u) for udp connections.

netstat -p | less: to see pid (-t) (-u) for udp connections.

netstat -n | less: to see numeric ports

netstat -c | less: it shows status continuous.

netstat -an | grep “:80”: to print the connections which contains port 80.

32. Ping:

The `ping` command is a network utility used to test the reachability of a host (e.g., a computer or server) on a network and measure the round-trip time for messages sent from the origin to the destination and back. It is often used to troubleshoot network connectivity issues.

How `ping` Works

- The `ping` command sends a series of Internet Control Message Protocol (ICMP) Echo Request packets to the target host.

- The target host, if reachable, responds with ICMP Echo Reply packets.

- The `ping` command then displays the results, including the time it took for the round-trip (in milliseconds) and any packet loss.

Basic Syntax

```
ping [options] destination
```

Common Usage Examples:

1. Ping a Domain or IP Address:

- To ping a website or IP address, simply type `ping` followed by the domain name or IP address.

```
ping google.com
```

- This will send ICMP Echo Request packets to `google.com` and display the results.

2. Ping a Specific Number of Times:

- Use the `-c` option to limit the number of ping requests sent. For example, to send 4 ping requests:

```
ping -c 4 google.com
```

3. Ping Indefinitely (Until Stopped):

- By default, `ping` on Unix-based systems will continue to send packets until it is stopped manually (usually with `Ctrl + C`)

```
ping google.com
```

- On Windows, `ping` sends 4 packets by default unless you specify otherwise with the `-t` option to keep pinging until manually stopped.

```
ping -t google.com
```

4. Specify Packet Size:

- You can specify the size of the packets sent using the `-s` option (Unix/Linux) or `-l` option (Windows). For example, to send packets of 1000 bytes:

```
ping -s 1000 google.com
```

5. View IP Address Only:

- Use the `-n` option to display the IP address of the host instead of the domain name (Unix/Linux).

```
ping -n google.com
```

Output Explanation:

- 64 bytes from 142.250.72.46: This indicates that 64 bytes of data were received from the IP address `142.250.72.46`.
- time=18.3 ms: This shows the round-trip time it took for the packet to travel from your machine to the target and back.
- TTL=115: The Time to Live (TTL) value indicates the number of hops (routers) the packet can pass through before being discarded.

Example Output

```
PING google.com (142.250.72.46): 56 data bytes
64 bytes from 142.250.72.46: icmp_seq=0 ttl=115 time=18.3 ms
64 bytes from 142.250.72.46: icmp_seq=1 ttl=115 time=18.5 ms
64 bytes from 142.250.72.46: icmp_seq=2 ttl=115 time=18.6 ms
```

Use Cases

- Check Network Connectivity: Quickly determine if a host is reachable over the network.
- Measure Latency: Gauge the time it takes for a packet to travel to a host and back.
- Troubleshoot Network Issues: Identify network delays, packet loss, or unreachable hosts.

The `ping` command is a fundamental tool in network administration and troubleshooting, providing immediate feedback on the status of a network connection.

33. Zip Command

The zip command is used to create compressed ZIP archives. A ZIP archive can contain multiple files and directories, and it reduces the file size by compressing the contents.

Basic Syntax

```
zip [options] archive_name.zip file1 file2 directory/
```

Common Usage Examples

1. Create a ZIP Archive:

- To compress multiple files into a single ZIP archive:

```
zip archive_name.zip file1.txt file2.txt
```

- This will create an archive named `archive_name.zip` containing `file1.txt` and `file2.txt`.

2. Include Directories:

- To include directories and their contents in the ZIP archive:

```
zip -r archive_name.zip directory_name/
```

- The -r (recursive) option ensures that all files and subdirectories within `directory_name/` are included.

3. Compress Files with Different Extensions:

- You can also compress all files of a certain type, for example, all `.txt` files in a directory:

```
zip text_files.zip *.txt
```

4. Add Files to an Existing ZIP Archive:

- To add files to an existing ZIP archive:

```
zip archive_name.zip newfile.txt
```

- This adds newfile.txt to `archive_name.zip` without affecting the existing contents.

34. unzip Command

The unzip command is used to extract the contents of a ZIP archive.

Basic Syntax

```
unzip [options] archive_name.zip
```

Common Usage Examples

1. Extract a ZIP Archive:

- To extract all the files and directories from a ZIP archive:

```
unzip archive_name.zip
```

- This will extract the contents of `archive_name.zip` into the current directory.

2. Extract to a Specific Directory:

- To extract the contents of a ZIP archive to a specific directory:

```
unzip archive_name.zip -d /path/to/destination/
```

- The `-d` option specifies the directory where the files should be extracted.

3. List the Contents of a ZIP Archive:

- To view the list of files in a ZIP archive without extracting them:

```
unzip -l archive_name.zip
```

4. Extract Specific Files:

- To extract specific files from a ZIP archive:

```
unzip archive_name.zip file1.txt file2.txt
```

- This will only extract `file1.txt` and `file2.txt` from the archive.

5. Overwrite Existing Files:

- By default, `unzip` will prompt you before overwriting any existing files. To automatically overwrite files without prompting:

```
unzip -o archive_name.zip
```

Examples

- Creating a ZIP archive with a directory:

```
zip -r my_archive.zip my_directory/
```

- Extracting a ZIP archive:

```
unzip my_archive.zip
```

- Extracting a ZIP archive to a specific directory:

```
unzip my_archive.zip -d /home/vipul/extracted_files/
```

35. Traceroute:

The `traceroute` (or `tracert` on Windows) command is a network diagnostic tool used to trace the path that packets take from your computer to a destination host (e.g., a website or server). It helps identify the route and measure the transit delays of packets across an IP network.

How `traceroute` Works:

- Path Discovery: `traceroute` determines the route taken by packets across the network by sending packets with progressively increasing Time-to-Live (TTL) values. Each router along the path decreases the TTL value by one before forwarding the packet. When the TTL reaches zero, the router sends back an ICMP "Time Exceeded" message, which `traceroute` uses to determine the address and response time of each hop.
- Latency Measurement: The command measures the round-trip time for each hop by sending multiple packets and reporting the time it takes for responses to return.

Basic Syntax

```
traceroute [options] destination
```

Common Usage Examples

1. Trace the Path to a Domain or IP Address:

- To trace the route to a specific domain or IP address:

```
traceroute google.com
```

- This will display each hop that packets take to reach `google.com`.

2. Trace with a Specific Number of Hops:

- To limit the number of hops:

```
traceroute -m 10 google.com
```

- The `-m` option specifies the maximum number of hops (in this case, 10).

3. Specify a Different Port Number:

- To use a different port number (useful for tracing specific services):

```
traceroute -p 80 google.com
```

- The `-p` option specifies the port number (80 for HTTP).

4. Use a Specific Protocol:

- To use UDP (default) or TCP for tracing:

```
traceroute -T google.com
```

- The `-T` option uses TCP instead of the default UDP.

Example Output

```
traceroute to google.com (142.250.72.206), 30 hops max, 60 byte packets
 1  router.local (192.168.1.1)  1.123 ms  1.065 ms  1.014 ms
 2  10.10.10.1 (10.10.10.1)  5.345 ms  5.296 ms  5.254 ms
 3  203.0.113.1 (203.0.113.1)  10.567 ms  10.455 ms  10.345 ms
 4  142.250.72.206 (142.250.72.206)  20.678 ms  20.565 ms  20.456 ms
```

Explanation of Output

- Hop Number: Indicates the order of hops (routers) in the path.
- IP Address/Hostname: Shows the IP address and (if available) the hostname of each hop.
- Round-Trip Times: The time it takes for packets to travel to the hop and back, usually reported for three packets.

tracert on Windows

On Windows systems, the command is `tracert` instead of `traceroute`. The usage is similar:

```
tracert google.com
```

Use Cases

- Network Troubleshooting: Identify where delays or issues occur in the network path.
- Diagnose Routing Problems: Determine if packets are being misrouted or dropped.
- Measure Latency: Assess the time it takes for packets to travel between nodes in the network.

traceroute (or `tracert`) provides valuable insights into network paths and performance, helping diagnose and resolve connectivity issues.

36. MTR:

The mtr (My Traceroute) command is a network diagnostic tool that combines the functionality of `traceroute` and `ping`. It provides a real-time view of the route packets take through the network and continuously monitors the performance of each hop along the path. mtr offers a more detailed and dynamic analysis compared to `traceroute` and `ping` alone.

How mtr Works

- Combination of traceroute and ping: `mtr` performs a continuous trace of the network path and measures latency and packet loss for each hop. It updates its display in real-time, allowing you to see the route and performance metrics dynamically.

- Dynamic Updates: Unlike `traceroute`, which provides a snapshot of the route at a single moment, `mtr` continuously updates its output to reflect changes in the network conditions.

Basic Syntax

```
mtr [options] destination
```

Common Usage Examples

1. Basic MTR Command:

- To start an `mtr` session to a specific domain or IP address:

```
mtr google.com
```

- This command will display the route and performance metrics to `google.com`.

2. Display Report in a Text-Based Format:

- To get a summary report of the mtr session (similar to `traceroute` output):

```
mtr --report google.com
```

3. Specify Number of Pings per Hop:

- To control the number of pings sent to each hop:

```
mtr -c 10 google.com
```

- The `-c` option specifies the number of pings (in this case, 10) to each hop.

4. Use a Specific Network Protocol:

- To use ICMP Echo (default), UDP, or TCP (depending on the implementation):

```
mtr --udp google.com
```

5. Save Output to a File:

- To save the output to a file for later review:

```
mtr google.com > mtr_output.txt
```

Example Output

Here's an example of what the `mtr` output might look like:

```
 My Traceroute  [v0.92]
 hostname: google.com
 DNS: google.com
 IP: 142.250.72.206
   Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
  Loss/Avg/Best/Worst/Mdev = 0/20.4/19.5/21.1/0.5 ms
   |
   1. 192.168.1.1 (192.168.1.1)  1.2 ms  1.0 ms  0.8 ms
   2. 10.10.10.1 (10.10.10.1)  5.6 ms  5.5 ms  5.4 ms
   3. 203.0.113.1 (203.0.113.1)  12.8 ms  12.9 ms  12.7 ms
   4. 142.250.72.206 (142.250.72.206)  22.3 ms  22.1 ms  22.0 ms
```

Explanation of Output

- Hop Number: Indicates the order of hops.
- Hostname/IP Address: Shows the hostname and IP address of each hop.
- Round-Trip Times: Times for packets to travel to each hop and back, usually shown for multiple pings.
- Loss/Avg/Best/Worst/Mdev: Packet loss percentage, average round-trip time, best time, worst time, and mean deviation.

Use Cases

- Network Performance Monitoring: Continuously track network performance and detect issues.

- Diagnose Connectivity Problems: Identify where delays or packet loss occur along the network path.

- Real-Time Analysis: See dynamic changes in network performance.

mtr is a powerful tool for network diagnostics, providing a continuous and comprehensive view of the route and performance of network packets.

37. AWK:

The `awk` command in Bash is a powerful text-processing utility that allows you to manipulate and analyze text files or streams. It is especially useful for pattern matching, extracting, and processing data based on certain conditions.

Basic Syntax

awk 'pattern { action }' [file ...]

- pattern: A condition or regular expression that determines which lines of the input are processed.

- action: A set of commands that are applied to each line that matches the pattern.

- file: The file(s) to process. If no file is provided, `awk` reads from standard input.

Basic Example:

awk '{ print $1 }' file.txt

- This command prints the first field (`$1`) of each line in `file.txt`. Fields are typically separated by spaces or tabs.

Common `awk` Commands and Options

1. Print Specific Fields:

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

- Prints the first and third fields of each line in `file.txt`.

2.Specify a Field Separator:

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

- The `-F` option sets the field separator (in this case, a comma `,`). This is useful for processing CSV files.

3. Pattern Matching:

```
awk '/pattern/ { print $0 }' file.txt
```

- Prints lines that match the specified pattern. `$0` refers to the entire line.

4. Sum a Column of Numbers:

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

- Sums up the values in the first column and prints the total after processing all lines.

5. Conditional Processing:

```
awk '$3 > 100 { print $1, $3 }' file.txt
```

- Prints the first and third fields only for lines where the value in the third field is greater than 100.

6. Use Built-in Variables:

- `NR`: The current line number.
- `NF`: The number of fields in the current line.

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

- Prints the line number and the entire line.

7. Custom Output Format:

```
awk '{ printf "Name: %s, Age: %d\n", $1, $2 }' file.tx
```

- Uses `printf` for formatted output. `%s` is for strings, and `%d` is for integers.

8. Replace Text in a Field:

```
awk '{ gsub(/old/, "new", $1); print }' file.txt
```

- Replaces occurrences of "old" with "new" in the first field of each line.

Examples

1. Extract the Usernames from `/etc/passwd`:

```
awk -F: '{ print $1 }' /etc/passwd
```

- The `-F:` option sets the field separator to a colon, which is used in the `/etc/passwd` file.

2. Print the Last Field of Each Line:

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

- `$NF` represents the last field in the line.

3. Filter Lines Based on Length:

```
awk 'length($0) > 20' file.txt
```

- Prints lines longer than 20 characters.

Summary

`awk` is a versatile command that can be used for simple tasks like extracting columns from a file or complex text processing involving pattern matching and conditional logic. It's a key tool for shell scripting and data processing in Unix-like systems.

38. SED:

The `sed` command in Bash is a stream editor used for parsing and transforming text in a pipeline. It's particularly useful for tasks like text substitution, deletion, and insertion.

Basic Syntax:

sed [OPTION]... 'SCRIPT' [FILE]...

- `OPTION`: Optional flags to modify `sed` behavior.

- `SCRIPT`: A set of instructions or commands for `sed` to execute.

- `FILE`: The file(s) to process. If no file is specified, `sed` reads from standard input.

Basic Examples

1. Substitute Text:

```
sed 's/old/new/' file.txt
```

- Replaces the first occurrence of "old" with "new" in each line of `file.txt`.

2. Substitute All Occurrences:

```
sed 's/old/new/g' file.txt
```

- Replaces all occurrences of "old" with "new" in each line of `file.txt`.

3. In-Place Editing:

```
sed -i 's/old/new/g' file.txt
```

- Replaces all occurrences of "old" with "new" directly in `file.txt` (in-place editing). The `-i` option edits the file in place.

4. Delete Lines Matching a Pattern:

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

- Deletes lines that match the specified pattern.

5. Print Specific Lines:

```
sed -n '2,4p' file.txt
```

- Prints lines 2 through 4 of `file.txt`. The `-n` option suppresses automatic printing, and `p` instructs `sed` to print the specified lines.

6. Insert Text Before a Line:

```
sed '2i\New line of text' file.txt
```

- Inserts "New line of text" before line 2 in `file.txt`.

7. Append Text After a Line:

```
sed '2a\New line of text' file.txt
```

- Appends "New line of text" after line 2 in `file.txt`.

8. Change a Line:

```
sed '2c\New content for line 2' file.txt
```

- Replaces line 2 with "New content for line 2".

Advanced Usage:

1. Use Multiple Commands:

```
sed -e 's/old/new/g' -e '2d' file.txt
```

- Applies multiple commands: replaces "old" with "new" and deletes line 2.

2. Use `&` to Reference the Matched Text:

```
sed 's/old/&_new/' file.txt
```

- Replaces "old" with "old_new", where `&` represents the matched text.

3. Use Regular Expressions:

```
sed -r 's/[0-9]+/number/g' file.txt
```

- Uses extended regular expressions (with `-r`) to replace sequences of digits with "number".

Summary

`sed` is a powerful tool for text processing and can handle a variety of text manipulation tasks efficiently. It's widely used in shell scripting and command-line operations for its ability to process and transform text streams in a non-interactive way.

Enabling SSH on ubuntu or your system:

Sudo apt install openssh-server: to install open ssh server on server

Config file is present at /etc/ssh/ssh_config where you can change the port.

SCP: Secure copy

Some time you may wonder In Unix, how do I securely transfer files between two Linux computers? So generally we use scp command to Securely Transfer Files/Folders in Linux (e.g. Ubuntu, Cent OS, Fedora, Linux mint ..). scp stands for "secure copy."

To copy from local to remote:

From 1 to 2

scp -r filename hostname@ipaddress destination location

scp -r dirname hostname@ipaddress destination location

to copy 2 to 1 like reverse:

scp hostname@ipaddress/source path (files/dir) /destination path

Installing Atom Editor in Ubuntu linux:

Atom is opensource source code and text editor. Atom can be installed on Windows, Linux and OS X. Atom supports plugins written in Node.js and has embedded Git source control. Atom is developed by GitHub. Atom is build using web technologies and used as a desktop application. In this post we will see how to install Atom editor on your Ubuntu Linux system.

```
$ sudo add-apt-repository ppa:webupd8team/atom

$ sudo apt-get update

$ sudo apt-get install atoms
```