The file contains only the 50 commands given in the pdf, I have described some more commands while doing the questions

41. ***man***

This returns a manual description of the command passed as argument to it, the **man** or rather the manual to the command contains all proportions of detailed information of the command
The **man** is the best option to get to know more about any command.
There can exist multiple **man** pages for the same command, we can access this passing the argument of page number to the command.
The details whether the command has multiple pages and what these pages are can be answer from the **whatis** command,
whatis <command_name>
This will give the one-line description from the **man** page(s) along with the number from where the command is taken from the **man**.
The manual so got from the above can be too long so for precise summary do the following:
<command>--help
This gives a shorter summary of the command.

50. ***wget***

According to the **man** page is the "the non-interactive network downloader".
wget is non-interactive, meaning that it can work in the background, while the user is not logged on, which is not usually possible as browsers require user presence to download from a website.
The network downloader means that the **wget** downloads from the network. The most common misunderstanding is **wget** downloads from the internet or world wide web, but this is not the case, **wget** downloads from a network, even if we set up a local network it can download from there if there exists a server to give it something in return to download or, in case of local network, copy.
The command downloads the given URL and saves it in the **pwd**. To download the **wget** uses certain protocols of the network namely HTTPS, FTP, HTTP. **wget** following these tries to get the data in a local html without redirections. The necessity to point out without redirection arose from the fact that **wget** retrieves is without the help of browser cache and logging history this implies that if you wish to download a file from your email then **wget** will fail you as it can't download from a secure page that requires login details.
The various options available in the wget include:

Startup:
-V, version-display the version of wget and exit
-b: background go to background after startup

Logging and input file:

-o: output-file=FILE log messages to FILE
-a: append-output=FILE append messages to FILE
-q: quiet quiet (no output)
-v: verbose be verbose (this is the default)
-i: input-file=FILE download URLs found in local or external FILE
-F: force-html treat input file as HTML
-B: base=URL resolves HTML input-file links (-i-F) relative to URL

Download:
-O: output-document=FILE write documents to FILE
-w: wait=SECONDS wait SECONDS between retrievals waitretry=SECONDS

The other similar command is **curl**.

49. ***date***

The date command does exactly what it says, it returns the instant the command was executed. The components of the output include day of week, day of month, month, hours of day, minutes in the hour and seconds in minutes, timezone and year respectively in a default format:
DAY MONTH DD HH:MM:SS <TimeZone(IST / UTC)> YYYY
There is a way to format the output date string, that is to place output control symbols after the date requested.
%a: locale's abbreviated weekday name (e.g., Sun)
%A: locale's full weekday name (e.g., Sunday)
%b: locale's abbreviated month name (e.g., Jan)
%B: locale's full month name (e.g., January)
%c: locale's date and time (e.g., Thu Mar 3 23:05:25 2005)
There are many more which can be viewed in short in the *date--help*.

48. ***ping***

ping is a computer network administration software utility used to test the reachability of a host on an Internet Protocol network. It is available for virtually all operating systems that have networking capability, including most embedded network administration software.
ping can be used to check the packet loss in transmission between local server and the server that is requested in the argument.
ping has many options, from changing the interval between pings to changing the format of address given URL, IPv4, IPv6, to changing the packet size sent, even limiting the number of packets sent / received.
Note: **ping** is used to check communication between servers not websites.

ping <address_in_some_format>

47. ***rpm***

rpm or the **R**ed-Hat **P**ackage **M**anager is one of the many package managers supported by various versions of Linux. A package manager is generally used to easily import external packages and their dependencies, it also allows the user to export a created package to others through the package manager.

46. ***apt***

(Note: The command given was **yum** but my Linux uses **apt**.)
man apt: “apt provides a high-level command line interface for the package management system. It is intended as an end user interface and enables some options better suited for interactive usage by default compared to more specialized APT tools. “
The **apt** tools include installation, updation, removal and even upgradation of the packages and their dependencies.
Usually the task done by **apt**, updating, installing or removing of packages, is not allowed to anyone but the root hence the **sudo** statement is attached to the command.
The **apt** is the default package manager in the Debian family.
The **yum** is the default package manager in the Red Hat family.
The command for basics of apt include:
For updating all getting the source list for all the packages.
apt-get update
For updating the packages in the updated source list.
apt-get upgrade
For removal of a particular package.
apt remove <package name>
For installation of particular packages.
apt install <package name>
Naturally the above commands require root access or it is either being used in root mode or using **sudo** before every command.

45. ***mysql***

mysql is the most commonly used database management system, more so in Linux systems.
Mysql is an open source database management system, providing the user to write in SQL (Structured Query Language).
Mysql requires a localhost server to host the database and the command **mysql** is used to login to the servers using the user access.
The usual working with **mysql** command is**-u** for user**-p** for password and**-h** for host name.

44. ***su***

The **su** command is used to access the root state of the terminal, here every command executed by the terminal is as the root i.e there is no permission denied possible.
The options in the **su** command are:
-c: (--command) COMMAND pass COMMAND to the invoked shell
-h: (--help) display this help message and exit
-l: (--login) make the shell a login shell
-p: (--preserve-environment) do not reset environment variables, and keep the same shell
The **su** is needed only to alter the files in the root directory, where the system files exist. Even in this case the use of **su** is recommended to be avoided so as to prevent changing files or dependencies and prevent doing damage to the system working.
To prevent casual use of **su** for installation or updation of packages, etc. the command **sudo** was introduced.
The **sudo** can be considered as root do**,** this allows the command to work on root privileges, though the command doesn't require root login only a root level access allowed to any user.

43. ***less***

There are files, commands, etc that have a very large output and to view them it is more convenient to view them page wise, the command used for this purpose we use the **less** command.
This command allows the user to view the information requested by the user in a new interface, this interface gives the user the ability to view information page wise and the movement to the next page and previous page is done using Ctrl+D and Ctrl+B.
The fun fact here is that **less** is used more than it's ancestor. **more** is used to get the display in the current shell session instead of opening an interface.
The command is usually more used with pipes because this takes the input from other command stdout.
<other_command_execution> | less

40 . ***locate***

The command does exactly what it says, it is used to search through the given path and it's subdirectories to look for the given pattern passed. The search can be further specified by giving arguments for file type and file name.
The options in the allocate command include:
-A: only print entries that match all patterns
-b: match only the basename of path names
-c: only print number of found entries
-e: only print entries for currently existing files

-h: help print this help
-i: ignore-case ignore case distinctions when matching patterns
-p: ignore-spaces ignore punctuation and spaces when matching patterns
-0: null separate entries with NULL on output
-q: quiet report no error messages about reading databases
-r: regexp REGEXP search for basic regexp REGEXP instead of patterns
-V: version print version information
-w: wholename match whole path name (default)
locate option pattern

39. ***whatis***

The command is used to get a brief description of the command, what it returns is the one-line description given in the **man** page of the command.
Some of the options in **whatis** are:
-v: print verbose warning messages
-r: interpret each keyword as a regex
-w: the keyword(s) contain wildcards
whatis <command_name>

38. ***whereis***

The **whereis** command is used to find the location of the command's executable in the root directory, this command also returns the location of the **man** pages of the command.
whereis <command_name>

37. ***uname***

The command gives information about the system and the kernel used.
-a: all print all information, in the following order, except omit-p and-i if unknown:
-s: kernel-name print the kernel name
-n: nodename print the network node hostname
-r: kernel-release print the kernel release
-v: kernel-version print the kernel version
-m: machine print the machine hardware name
-p: processor print the processor type (non-portable)
-i: hardware-platform print the hardware platform (non-portable)
-o: operating-system print the operating system
--help display this help and exit
--version output version information and exit

uname <OPTION>

36. ***ifconfig***

The command is used to configure the network interface that is used in the current system.
The configuration varies greatly from masking an IP to enabling or disabling use of protocols, etc.
-a: display all interfaces which are currently available, even if down
-s: display a short list (like netstat-i)
-v: be more verbose for some error conditions

35. ***mkdir***

This is used for making a new directory in the **pwd**. The name is passed as the argument to the command.

34. ***passwd***

Change your password from command line using **passwd**. This will prompt for the old password followed by the new password.
Superuser can use **passwd** command to reset others password. This will not prompt for the current password of the user.
passwd USERNAME
If root access and forcefully change non-root password then:
passwd-d USERNAME

33. ***chown***

The command is used to change the owner of the file that is given, the owner change doesn't mean that the group that owns it will change, we have to input the other group name too.
chown owner:group filename
The options in the commands include:
-c:--changes like verbose but report only when a change is made
-f:--silent:--quiet suppress most error messages
-v:--verbose output a diagnostic for every file processed
-R:--recursive operate on files and directories recursively

32. ***chmod***

The command is used to change read, write and execute permissions of the file. The file permissions have 3 * 3 bits allocated to them, where the 3-bits are for rwx each for owner, group and other group, respectively.
The binary representation of the above 3-bits for each stakeholder can be used as input to change file permissions.
Eg.
chmod 666 <filename>
The above will give read and write permissions to all three.
We can also give permission to user(**-u**), group (**-g**) and other group(**-og**) by equating the permission to respective binary or the file permission that would be granted
Eg.
chmod-u=5 <filename>
and
chmod-u=rx <filename>
The above two do the same, give the user read and execute permissions.

31. ***mount***

The **man** says
"
All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the umount(8) command will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by a network or other service.
"
Before a user can access a file on a Unix-like machine, the file system that contains it needs to be mounted with the mount command. Frequently mount is used for SD card, USB storage, DVD and other removable storage devices.
The **mount** is usually used on external devices so these are visible /dev.
mount /dev/ext_device_name /mnt

30. ***cat***

The **man** page says
"concatenate files and print on the standard output"

The cat command used as:
Display file:
cat filename
Write/Overwrite ‘file2’:
cat file1 > file2
The above command will copy the contents from ‘file1’ to ‘file2’, if the ‘file1’ argument parameter is left blank then the default ‘file1’ stream is taken as stdin, if the ‘file2’ then there is an error.
Append to file2:
cat file1 >> file2
The ‘file1’ contents are appended at the end of ‘file2’. Similar to writing using **cat** if the ‘file1’ is left blank then the stdin is taken.

29. ***mv***

The command is used to move file(s) in the file system. This can be used to rename a file if we put the source path and destination path as the same.
mv source_path dest_path
The options in the command include:
-t:--target-directory=DIRECTORY move all SOURCE arguments into DIRECTORY
-T:--no-target-directory treat DEST as a normal file
-u:--update move only when the SOURCE file is newer than the destination file or when the destination file is missing
-v:--verbose explain what is being done

28. ***cp***

The **cp** command is used to copy the file from source file path to destination file path.
cp source_path dest_path

27. ***rm***

The **rm** command is used to remove the file(s) from the file system.
rm-<option> <file/filepath>
The options under this command include:
-f:--force, ignore nonexistent files and arguments, never prompt
-i:--interactive, prompt before every removal
-r:-R:--recursive, remove directories and their contents recursively
-d:--dir, remove empty directories

26. ***kill***

The command is used to pass signals to the process. The signal can be varied like
9: terminate the process.
kill-<signal> <process_id>

25. ***df***

The man says df is used to “report file system disk usage”.
The file system that is existing needs disk space for processing, execution or even display of data, the usage of the **df** command returns the details related to this.
Namely the file system used, % process is using, number of blocks(available, used, free) and where the file is mounted.
df <file(s)>
If no arguments are passed to then the space available on all currently mounted file systems is shown.

24. ***top***

The command is used to display the top processes in the system, sorted by the options given in command.
These options include:
a: PID = Process Id
v: nDRT = Dirty Pages count
d: UID = User Id
y: WCHAN = Sleeping in Function
e: USER = User Name
z: Flags = Task Flags

23. ***free***

This command is used to “display the amount of free and used memory in the system”.
free displays the total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel. The information is gathered by parsing /proc/meminfo.
The displayed columns are from the information in /proc/meminfo/:

Total: Total installed memory
Used: Used memory (calculated as total- free- buffers- cache)
Free: Unused memory
Shared: Memory used by tmpfs
Buffers: Memory used by kernel buffers
Cache: Memory used by the page cache and slabs
buff/cache: Sum of buffers and cache available estimation of how much memory is available for starting new applications, without swapping.

22. ***ps***

The **ps** gives information about processes running.
The various options in **ps** allow the user to select which process to be displayed, by default the processes displayed are the process in the current terminal session.
The options are:
-A: Select all processes.Identical to-e.
-a: Select all processes except both session leaders (see getsid(2)) and processes not associated with a terminal.
-d: Select all processes except session leaders.
-e: Select all processes.Identical to-A.
-N: Select all processes except those that fulfill the specified conditions (negates the selection).Identic

The process tree is displayed by:
ps-ejH

21. ***service***

The job of starting, stopping, or reloading this binary application is handled by the service's init script. It's called the init script because it initializes the service. In System V, an init script is a shell script. Init scripts are also called rc (run command) scripts.
Unix System V is one of the first commercial versions of the Unix operating system.
Usage:
service [service] [option]
The **service** command runs the above mentioned System V init script.
This will return all the services running:
service--status-all
The services listed above can now be used as the**[service]**.
The options in the services are **{start|stop|graceful-stop|restart|reload|force-reload}**.

20. ***crontab***

The **Cron** daemon is a built-in **Linux** utility that runs processes on your system at a scheduled time. **Cron** reads the **crontab** (**cron** tables) for predefined commands and scripts. By using a specific syntax, you can configure a **cron** job to schedule scripts or other commands to run automatically. (from phoenixnap.com due to m).

19. ***ftp***

The **ftp** is used to download files from a remote location..
ftp IP/hostname
The hostname or an IP address is passed to the ftp and we can download the information that the IP/host is sending us.
Options may be specified at the command line, or to the command interpreter.
-4: Use only IPv4 to contact any host.
-6: Use IPv6 only.
-i: Turns off interactive prompting during multiple file transfers.
There is also **sftp** for secure ftp usage.

18. ***shutdown***

The command shutdowns the system. There are various options available to this command, they include:
-H: Halt the machine
-P:Power-off the machine
-r:Reboot the machine
-h: Equivalent to--poweroff, overridden by--halt
-k: Don't halt/power-off/reboot, just send warnings
-c: Cancel a pending shutdown

17. ***unzip***

The command to extract data from the zip file.
The best use of the command is given in **man** page:
“
Usage: unzip [-Z] [-opts[modifiers]] file[.zip] [list] [-x xlist] [-d exdir]

Default action is to extract files in list, except those in xlist, to exdir file[.zip] may be a wildcard.-Z => ZipInfo mode ("unzip-Z" for usage).
-pextract files to pipe, no messages :-llist files (short format)
-ffreshen existing files, create none:-ttest compressed archive data
-uupdate files, create if necessary:-zdisplay archive comment
-xexclude files that follow (in xlist):-dextract files into exdir

modifiers:: :-qquiet mode (-qq => quieter)
-nnever overwrite existing files :-aauto-convert any text files
-ooverwrite files WITHOUT prompting:-aa treat ALL files as text
-jjunk paths (do not make directories) :-vbe verbose/print version info
-Cmatch filenames case-insensitively:-Lmake (some) names lowercase
"

The file can be compressed (in zip) using **zip** command

16. ***bzip2***

The command is used to make the list of files.
The options in the command include:
-d:--decompress force decompression
-z:--compress force compression
-k:--keep keep (don't delete) input files
-f:--force overwrite existing output files
-t:--test test compressed file integrity
-c:--stdout output to standard out
-q:--quiet suppress noncritical error messages
-s:--small use less memory (at most 2500k)
-1: set block size to 100k
-9: set block size to 900k
The is used to zip and unzip but if we use **bunzip2** this will directly unzip the file.

15. ***gzip***

The command is used to zip and unzip files.
The options in the command include:
-c:--stdout write on standard output, keep original files unchanged
-d:--decompress decompress
-f:--force force overwrite of output file and compress links
-l:--list list compressed file contents

-q:--quiet suppress all warnings
-r:--recursive operate recursively on directories
-S:--suffix=SUF use suffix SUF on compressed files
-t:--test test compressed file integrity
-v:--verbose verbose mode

14. ***cd***

The command is used to change directories in the file system.
cd DIR
Change the current directory to DIR. The default DIR is the value of the HOME shell variable.

13. ***pwd***

The **pwd** returns the name of the present working directory of the terminal session in which it is requested.

12. ***ls***

The command is used to list the files present in a given path.
The options of **ls** include:
-l: details of the list
-h: give the details of-l in human readable form
-R: recursively display files
-d: display only the list of directories
-n: like-l, but uses group and user IDs
-a: display all files, even hidden

11. ***xargs***

The command does is execute arguments.
The arguments in this are space separated and hence it is wise to write the command within “”.
The output of the command is ignored (not displayed on stdout), the commands do there work but the stdout is not allowed.
The **xargs** command contains various options:
-d:--delimiter, to change the delimiter from the default space character
-a: read the input from input file passed as argument instead of stdin

10. ***export***

export is bash shell BUILTINS commands, which means it is part of the shell. It marks environment variables to be exported to child-processes.
Options in **export** include:
-f: refer to shell functions
-n: remove the export property from each NAME
-p: display a list of all exported variables and functions

9. ***sort***

The command is used to sort the lines in a file, in a given way as specified in the options given to the file.
The ordering options of **sort** include:
-b:--ignore-leading-blanks ignore leading blanks
-d:--dictionary-order consider only blanks and alphanumeric characters
-f:--ignore-case fold lower case to upper case characters
-g:--general-numeric-sort compare according to general numerical value
-i: consider only printable characters
-M:--month-sort compare (unknown) < 'JAN' < ... < 'DEC'
-h:--human-numeric-sort compare human readable numbers (e.g., 2K 1G)
-n:--numeric-sort compare according to string numerical value
-R:--random-sort shuffle, but group identical keys
-r:--reverse reverse the result of comparisons
-m:--merge merge already sorted files; do not sort
-o:--output=FILE write result to FILE instead of standard output
-s:--stable stabilize sort by disabling last-resort comparison

8. ***diff***

The command is used to differentiate between two files, the true problem in the command is understanding the output.
The interpretation of the **diff** output is a job in itself, this is explained with examples, in the questions document.

7. ***vim***

vim is one of the built-in command line based text editors available in Linux. The **vim** is now improved to **gvim** and many other versions of this editor.

6. ***awk***

The **awk** is a scripting language used to display contents of a file. **awk** is a scripting language and hence the way input is unique too.
awk <pattern> <replaced_pattern> filename
This is the general form of an **awk** command.
Due to **awk** being more than a command the options available are also more, the options available vary from printing the columns that are requested to replacing these contents to the original file.
awk unlike the usual command in Linux has its own interpreter. This could have caused inconsistencies if not for regular expressions.

5. ***sed***

Much similar to **awk**, the stream editor or **sed** changes the content of the file, here the file content is actually altered, hence the name stream editor.
sed much like **awk** has its own interpreter. The input can also be given in the form of regular expressions.

4. ***ssh***

The **ssh** command “OpenSSH SSH client” says the man page.
The ssh client is a program used to log into a remote machine for executing commands on the machine. It is intended to provide secure encrypted communications between two untrusted users over an insecure network.
The options include:
-4: Forces ssh to use IPv4 addresses only.
-6: Forces ssh to use IPv6 addresses only.
-A: Enables forwarding of the authentication agent connection. This can also be specified on a per-host basis in a configuration file.
-a: Disables forwarding of the authentication agent connection.
-c: cipher_spec, Selects the cipher specification for encrypting the session. cipher_spec is a comma-separated list of ciphers listed in order of preference.
-E: log_file, Append debug logs to log_file instead of standard error.
-G: Causes ssh to print its configuration after evaluating Host and Match blocks and exit.
-g: Allows remote hosts to connect to local forwarded ports. If used on a multiplexed connection, then this option must be specified on the master process.

(Due to constraints I have only tried ssh connection with github and that works just fine but I am unable to completely explore the command).

3. ***find***

The command is used to search files in the given file path. The command prints the name of the file that has the specifications given in the command input.
The **find** command is used as:
find <path> -<option> <expression>
The expression is entered in regular expression format.
The options here may include the specifications of the file we search:
-name: search the name of the file
-empty: search for empty directory or file
-perm: search for files with this permission

2. ***grep***

The command is used to find expressions inside given files.
The command is used as follow as:
grep <options> <pattern> <file>
The options include:
-c: This prints only a count of the lines that match a pattern
-h: Display the matched lines, but do not display the filenames.
-i: Ignores, case for matching
-l: Displays list of filenames only.
-n: Display the matched lines and their line numbers.
-v: This prints out all the lines that do not matches the pattern
-e: exp: Specifies expression with this option. Can use multiple times.
-f: file: Takes patterns from file, one per line.
-E: Treats pattern as an extended regular expression (ERE)
-w: Match whole word
-o: Print only the matched parts of a matching lin

1. ***tar***

The command is used to an archive file and unarchive the given file.
The options in tar include:
-c: Creates Archive
-x: Extract the archive
-f: creates archive with given filename
-t: displays or lists files in archive file
-u: archives and adds to an existing archive file
-v: Displays Verbose Information

-A: Concatenates the archive files
-z: zip, tells tar command that create tar file using gzip
-W: Verify a archive file
-r: update or add file or directory in already existed .tar file
The creation of tar archive is done using:
tar cvf <archieve_name> <list_of_files>
To view the files in an archive:
tar tf <archive_name>
To extract from a tar archive:
tar vf <archive_name>
The tar is one of the basic archive management tools used in Linux, and by far the most widely used one.