MAHARAJA SURAJMAL INSTITUTE

(Affiliated to Guru Gobind Singh Indraprastha University)

LINUX PRACTICAL FILE

Submitted By: Deepak Chauhan

02714902020

BCA 6B

INDEX

36.	What is the use of “gzip"&*gunzip” command?	
37.	What is the use of “bzip2” & “bunzip2” command?	
38.	What is the use of “zip?&*Unzip” command?	
39.	What is the use of “tar” command?	
40.	What is the use of “pr” command?	
41.	What is the use of “head” command?	
42.	What is the use of “tail” command?	
43.	What is the use of “paste” command?	
44.	What is the use of “sort” command?	
45.	What is the use of “uniq” command?	
46.	What is the use of “tr” command?	
47.	What is the use of “cmp” command?	
48.	What is the use of “comm” command?	
49.	What is the use of “diff” command?	
50.	What is the use of “aspell” command?	
51.	What is the use of “grep” command?	
52.	What is the use of “sed” command?	
53.	What is use of “history” command?	
54.	What is the use of tee command?	
55.	What is the use of umask?	
56.	What is the use of suid, sgid, sticky bit?	
57.	Ps command	
58.	Pstree command	
59.	Nice command	
60.	Top command	
61.	Calculate the area of a triangle	
62.	Calculate the area of a rectangle	
63.	Calculate the area of a square	
64.	Find the area of a circle	
65.	Find whether a number entered is even or odd	
66.	Find the status of the student according to the given	
67.	Find the status of a student according to given constraints	
68.	Find whether the year is leap or not	
69.	Find the largest number among three	
70.	Find the larger of two numbers	
71.	Convert the temperature Fahrenheit into Celsius	
72.	Illustrate case statement by making 12 cases each representing a month	
73.	Menu driven program using the arithmetic operators	

74.	Illustrate case statement defining a case for each [A-Z}, [2-7] and {0-9]	
75.	To illustrate date, Is, ps, time.	
76.	Calculate sum of first n natural numbers	
77.	Sort n numbers	
78.	Determine whether the given number is prime or not	
79	Find the factorial of a given number	
80.	Print the Fibonacci series	
81.	Display multiplication table of a number	
82.	Find the position of a word from the string	
83.	To count number of words, characters and blank space	
84.	What is the use of Redirection symbol?	
85.	What is the use of Standard Input, output, error?	
86.	What is the use of Pipe operator?	
87.	What is use of VI editor?	
88.	What is use of SU command?	

QUESTION 1:- WHAT IS USE OF "HELP" COMMAND?

Displays information about built in commands. Syntax: - help [-dms][PATTERN...]
Options:-

d = output short description for each topic.
-m = display usage in pseudoman page format.
-s = output only a short usage synopsis for each topic matching.

OUTPUT

```
~$ help pwd
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
                directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.
~$ help -d pwd
pwd - Print the name of the current working directory.
~$ help -m pwd
NAME
    pwd - Print the name of the current working directory.

SYNOPSIS
    pwd [-LP]

DESCRIPTION
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
                directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.

SEE ALSO
    bash(1)

IMPLEMENTATION
    GNU bash, version 5.0.17(1)-release (x86_64-pc-linux-gnu)
    Copyright (C) 2019 Free Software Foundation, Inc.
    License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

~$
```

QUESTION 2:- WHAT IS THE USE OF "MAN" COMMAND?

It is the interface used to view the system's reference manual.

Syntax: - man [-hV]
Options:-

-h,--help = Print a help message and exit.
-V,--version =Display version information and exit.
-d,--debug = Print debug information.

OUTPUT (man pwd)

```
PWD(1)

NAME
       pwd - print name of current/working directory

SYNOPSIS
       pwd [OPTION]...

DESCRIPTION
       Print the full filename of the current working directory.

       -L, --logical
              use PWD from environment, even if it contains symlinks

       -P, --physical
              avoid all symlinks

       --help display this help and exit

       --version
              output version information and exit

       If no option is specified, -P is assumed.

       NOTE: your shell may have its own version of pwd, which usually supersedes the versio

AUTHOR
       Written by Jim Meyering.

REPORTING BUGS
       GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
       Report pwd translation bugs to <https://translationproject.org/team/>

COPYRIGHT
       Copyright © 2018 Free Software Foundation, Inc.  License GPLv3+: GNU GPL version 3 or
       This is free software: you are free to change and redistribute it.  There is NO WARRA

SEE ALSO
       getcwd(3)

       Full documentation at: <https://www.gnu.org/software/coreutils/pwd>
       or available locally via: info '(coreutils) pwd invocation'

GNU coreutils 8.30
 Manual page pwd(1) line 1/44 (END) (press h for help or q to quit)
```

After h (help)

```
                  SUMMARY OF LESS COMMANDS

      Commands marked with * may be preceded by a number, N.
      Notes in parentheses indicate the behavior if N is given.
      A key preceded by a caret indicates the Ctrl key; thus ^K is ctrl-K.

  h  H                 Display this help.
  q  :q  Q  :Q  ZZ     Exit.
 ---------------------------------------------------------------------------

                           MOVING

  e  ^E  j  ^N  CR  *  Forward  one line   (or N lines).
  y  ^Y  k  ^K  ^P  *  Backward one line   (or N lines).
  f  ^F  ^V  SPACE  *  Forward  one window (or N lines).
  b  ^B  ESC-v      *  Backward one window (or N lines).
  z                 *  Forward  one window (and set window to N).
  w                 *  Backward one window (and set window to N).
  ESC-SPACE         *  Forward  one window, but don't stop at end-of-file.
  d  ^D             *  Forward  one half-window (and set half-window to N).
  u  ^U             *  Backward one half-window (and set half-window to N).
  ESC-)  RightArrow *  Right one half screen width (or N positions).
  ESC-(  LeftArrow  *  Left  one half screen width (or N positions).
  ESC-}  ^RightArrow   Right to last column displayed.
  ESC-{  ^LeftArrow    Left  to first column.
  F                    Forward forever; like "tail -f".
  ESC-F                Like F but stop when search pattern is found.
  r  ^R  ^L            Repaint screen.
  R                    Repaint screen, discarding buffered input.
        ---------------------------------------------------
        Default "window" is the screen height.
        Default "half-window" is half of the screen height.
 ---------------------------------------------------------------------------

                           SEARCHING

  /pattern          *  Search forward for (N-th) matching line.
  ?pattern          *  Search backward for (N-th) matching line.
  n                 *  Repeat previous search (for N-th occurrence).
  N                 *  Repeat previous search in reverse direction.
  ESC-n             *  Repeat previous search, spanning files.
  ESC-N             *  Repeat previous search, reverse dir. & spanning files.
  ESC-u                Undo (toggle) search highlighting.
  &pattern          *  Display only matching lines
        ---------------------------------------------------
        A search pattern may begin with one or more of:
        ^N or !  Search for NON-matching lines.
        ^E or *  Search multiple files (pass thru END OF FILE).
        ^F or @  Start search at FIRST file (for /) or last file (for ?).
        ^K       Highlight matches, but don't move (KEEP position).
HELP -- Press RETURN for more, or q when done
```

QUESTION 3:-WHAT IS THE USE OF "INFO" COMMAND?

It is similar to "man" with a more robust structure for linking pages together. The default location of info documentation is /usr/share/info.

Syntax: - info [OPTION]... [MENU-ITEM...]
Options:-
-k,--apropos = Look up STRING in all indices of all manuals.
-d,--directory = Add DIR to INFOPATH.
-f,--file = Specify Info file to visit.
-h,--help =Display this help and exit.
-o,--output =Output selected nodes to FILENAME. -version = Display version information and exit**.**

OUTPUT (info grep)

```
Next: Introduction,  Up: (dir)

grep
****

'grep' prints lines that contain a match for one or more patterns.

   This manual is for version 3.4 of GNU Grep.

   This manual is for 'grep', a pattern matching engine.

   Copyright © 1999–2002, 2005, 2008–2020 Free Software Foundation, Inc.

     Permission is granted to copy, distribute and/or modify this
     document under the terms of the GNU Free Documentation License,
     Version 1.3 or any later version published by the Free Software
     Foundation; with no Invariant Sections, with no Front-Cover Texts,
     and with no Back-Cover Texts.  A copy of the license is included in
     the section entitled "GNU Free Documentation License".

* Menu:

* Introduction::                Introduction.
* Invoking::                    Command-line options, environment, exit status.
* Regular Expressions::         Regular Expressions.
* Usage::                       Examples.
* Performance::                 Performance tuning.
* Reporting Bugs::              Reporting Bugs.
* Copying::                     License terms for this manual.
* Index::                       Combined index.
```

(info -a grep)

```
Node: File names matching 'grep'

Info File Index
***************

File names that match 'grep':

* Menu:

*     1: (/usr/share/info/grep.info.gz).
*     2: (*manpages*)grep.
```

(info -w grep)

```
~$ info -w grep
/usr/share/info/grep.info.gz
```

QUESTION 4:- WHAT IS THE USE OF "WHATIS" COMMAND?

This command searches the manual page name and display the manual page description of any matched.

Syntax: - whatis[-dlvV][-r|-w][-m system[,...]] [-M path] [-L local]

Options:-

-d = Print debugging warning information.

-v,--verbose= Print verbose warning messages.

-l,--long = Interpret each name as a regular expression.

-e,--extract = do not trim output to the terminal width.

Example: - $whatis whatis

OUTPUT

```
~$ whatis whatis
whatis (1)           - display one-line manual page descriptions
~$
```

QUESTION 5:- WHAT IS USE OF "TYPE" COMMAND? Display information about command type.

Syntax: - type [-afptP] name [name...]
Options:-
-a = display all locations containing an executable named NAME includes aliases.
-f = suppress shell function look up.

Example: - type mkdir

OUTPUT

```
~$ type mkdir
mkdir is /usr/bin/mkdir
~$ type -a mkdir
mkdir is /usr/bin/mkdir
mkdir is /bin/mkdir
```

QUESTION 6:-WHAT IS THE USE OF "W" COMMAND?

The w command is a quick way to see who is logged on and what they are doing

Syntax:- w (option) user (.....)

Option:-
-h,--no-header =Don't print the header.
-u, --no–current = Ignore the username while figuring out the current process and cpu . User =Shows information about the specified user only .

OUTPUT

```
~$ w
 12:20:53 up  5:26,  0 users,  load average: 0.85, 0.89, 0.98
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
~$ w -h
~$ w -o
 12:21:15 up  5:26,  0 users,  load average: 0.87, 0.90, 0.99
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
~$ w -V
w from procps-ng 3.3.16
~$ w -s
 12:21:37 up  5:27,  0 users,  load average: 0.79, 0.87, 0.98
USER     TTY      FROM              IDLE WHAT
~$
```

QUESTION 7:- WHAT IS THE USE OF "DATE" COMMAND? This command is used to display the current date and time.

Syntax: - date [OPTION]... [+FORMAT] Options:-
-d,--date =Display the current date and time.

Example: - $date +%m

OUTPUT

```
~$ date
Thu Apr 20 12:26:37 UTC 2023
~$ date +%Z
UTC
~$ date +%X
12:27:05
~$
```

QUESTION 8:- WHAT IS THE USE OF "BC" COMMAND?
BC is an arbitrary -precision language for performing math calculations. bc is a language

that supports arbitrary- precision numbers, meaning that it delivers accurate results regardless of how large the numbers are.

Syntax : **bc [-hlwsqv] [long-options] [file ...]**

Options:

Options:
-h, {- -help } : Print the usage and exit
-i, {- -interactive } : Force interactive mode
-l, {- -mathlib } : Define the standard math library
-w, {- -warn } : Give warnings for extensions to POSIX bc
-s, {- -standard } : Process exactly the POSIX bc language
-q, {- -quiet } : Do not print the normal GNU bc welcome
-v, {- -version } : Print the version number and copyright and quit

OUTPUT

```
~$ echo "12+5" | bc
17
~$ echo "var=10;var" | bc
10
~$
```

QUESTION 9:- WHAT IS USE OF "CAL" COMMAND?
Display a conventially - formatted calendar form the command line.

Syntax: -cal [options][[[day]month]year]
Options:-
-1 = display a single month. This is default.
-s = display the calendar using Sunday as the first day of the week. -j = display dates of the Julian calendar.

Example: - $cal 4 2015 OUTPUT

OUTPUT

```
~$ cal 5 2023
      May 2023
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

QUESTION 10:- WHAT IS USE OF "WHO" COMMAND? This command prints about all users who are currently logged in.

Syntax: - who [OPTION]... [FILE][ami]
Options:-
-a,--all = Same as using the options –b,-d—login,-p,-r,-t,-T,-u.
-b = Display the time of the last system boot.
-d,--dead = Display dead processes.
Example: - $who

-H command prints the column headers and when combined with the –u option provides a more detailed list.
Syntax: - $who -Hu

Example: - $who –Hu

OUTPUT

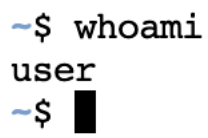
```
~$ who -Hu
NAME     LINE         TIME             IDLE          PID COMMENT
~$
```

QUESTION 11:-WHAT IS USE OF "WHOAMI" COMMAND? This command prints the effective user ID.

Syntax: -whoami [OPTION]
Options:-
--help = Display a help message and exit. --version = Display version information and exit.

Example: -whoami

OUTPUT



QUESTION 12:- WHAT IS THE USE OF "ECHO" COMMAND? This command is used to display the line of text.

Syntax: -echo[OPTION]
Options:-
-n = Do not output the trailing newline.
-e = Enable interpretation of backslash escapes. -E = Disable interpretation of backslash escapes.

Example: - $echo hello world

OUTPUT

```
~$ echo -e "hello \n world" after quotes
hello
 world after quotes
~$ echo hello world
hello world
~$ echo -e "hello \n world \t goodbye"
hello
 world   goodbye
~$
```

QUESTION 13:- WHAT IS USE OF “PRINTF” COMMAND?
printf inserts arguments into a user defined string or text, creating formatted output.

Syntax: -printf FORMAT [ARGUMENT]..
Options:-
FORMAT = it controls the output and defines the way that the arguments will be expressedin the output.

ARGUMENT = each ARGUMENT will be inserted into the formatted output according to the definition of FORMAT.

Example: -printf “MAHARAJA SURAJMAL” OUTPUT

OUTPUT

```
~$ printf "MAHARAJA SURAJMAL"
MAHARAJA SURAJMAL~$ printf "My current shell is %s \n" $SHELL
My current shell is /bin/bash
~$
```

QUESTION 14:-WHAT IS THE USE OF "TTY" COMMAND? It prints the file name of the terminal connected to standard input.

Syntax: -tty[OPTION]
Options:-
-s,--silent , --quite = Print nothing, only return an exit status.

Example: - $tty

OUTPUT

```
~$ tty
/dev/pts/1
~$
```

QUESTION 15:- WHAT IS USE OF "STTY" COMMAND?
This command is used for both displaying and changing the setting of your terminal.

Syntax: -stty[-F DEVICE]—file-DEVICE][SETTING] Options:-
-A,-ALL = Print all current setting in human readable form.
-g,--save = Print all current setting in a sty readable form.
-F,--file = Open a help message and exit.

Example: - $stty

OUTPUT

```
~$ stty
speed 38400 baud; line = 0;
eol = M-^?; eol2 = M-^?;
ixany iutf8
~$ stty -a
speed 38400 baud; rows 52; columns 214; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = M-^?; eol2 = M-^?; swtch = 
-parenb -parodd -cmspar cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc ixa
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echo
~$
```

QUESTION 16:- WHAT IS THE USE OF "PATH VARIABLE"? List of directories searched by shell to locate a command.

Example: - echo $PATH

OUTPUT

```
~$ echo $PATH
/cocalc/bin:/cocalc/src/smc-project/bin:/home/user/bin:/home/user/.local/bin:/ext/bin:/usr/lib/xpra:/opt/
ghc/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/ext/data/homer/bin:/ext/data/weblog
o:/usr/lib/postgresql/10/bin
~$
```

QUESTION 17:- WHAT IS USE OF "SHELL,USER,LANG, DISPLAY,HOME,TERM,LOGNAME, PS1,PS2, MAIL, MAILCHECK"?

SHELL-User's login shell and one invoked by programs having shell escapes.
HOME- This variable stores the login information of the user.
LOGNAME- login name of the user.
PS1- Primary Prompt String
Syntax: - $PS1="MYENV>"
PS2- Secondary Prompt String
Syntax:-$PS2='>'
MAIL- Absolute pathname of user's mailbox file Syntax- $MAIL
MAILCHECK- Mail Checking interval for incoming mail. Syntax:-echo $MAILCHECK
LANG-:-Used to specify to desired locale (national language and region) Syntax:-echo $LANG
TERM- It show the type of the terminal
USER- username
DISPLAY- instructs an X client which X server it is to connect to by default

Example: - echo $SHELL

OUTPUT

```
~$ echo $USER
user
~$ echo $SHELL
/bin/bash
~$ echo $LANG
en_US.UTF-8
~$ echo $DISPLAY

~$ echo $HOME
/home/user
~$ echo $TERM
xterm-256color
```

QUESTION 18:-WHAT IS USE OF "UNAME" COMMAND? Print information about the current system. Default option is –s.

Syntax: -uname[option]...

Options:-
-l,-all = Prints all information.
-s,--kernal-name = Print the kernel name.
-n,--nodename = Print the network mode name.

Example: - $uname –a

OUTPUT

```
~$ uname
Linux
~$ uname -n
project-6a305fe9-b9d7-452a-acd0-574e4b4b7789
~$ uname --all
Linux project-6a305fe9-b9d7-452a-acd0-574e4b4b7789 5.13.0-1033-gcp
4:12 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
~$
```

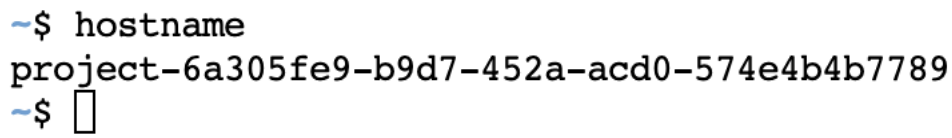
QUESTION 19:-WHAT IS THE USE OF “HOSTNAME” COMMAND?

Hostname is the name of the system or server you are logged into. The hostame can also refer to the sitename. As and example, if an organix=zation domain name is “google.com” na da specific computer name in that doman is “unix-box”, then the host name of the computer is “unix-box.google.com”. Syntax :- hostname [option]... [file]...

Options:-
-a =print the alisa name of the host if created any. -d =print the domain name
-I =prints the ip address of the host
-s =print the shortname of the host
-v =verbose data
-V =version information
-h =help about hostname command

Example: - $ hostname

OUTPUT



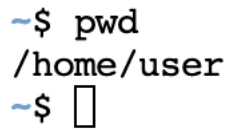
QUESTION 20:-WHAT IS USE OF "PWD" COMMAND?
This command prints the full pathname of the current working directory.

Syntax: -pwd [OPTION]

Options:-
-L,--logical = Used to display the logical address in the system.
-p,--physical = It print the fully resolved name for the current directory.
--help = Display a help message and exit.
--version = Display version information and exit.

Example: - $pwd OUTPUT

OUTPUT



QUESTION 21:-WHAT IS USE OF “CD” COMMAND?

Ans: The cd command which stands for “change directory”,changes the shell’s current working directory.
Syntax: - cd[-L|-P]directory

Options:-
-L = force symbolic links to be followed.
-P = use the physical directory structure without following symbolic links.
Example: - cd hope

OUTPUT

```
~$ cd hope
~/hope$
```

QUESTION 22:-WHAT IS USE OF "MKDIR" COMMAND? This command is used to create directories on a file system.

Syntax: -mkdir [option...] DIRECTORY...

Options:-
-m,--mode = Set file mode.
-p,--parents = Create parent directories as necessary.

Example: - $ mkdirmydir $mkdir-m a=rwxmydir

OUTPUT

```
~$ mkdir testDir
~$ cd testDir
~/testDir$ cd insideDir
bash: cd: insideDir: No such file or directory
~/testDir$ mkdir insideDir
~/testDir$ cd ..
~$ cd testDir
~/testDir$ cd insideDir
~/testDir/insideDir$ cd ..
~/testDir$ cd ..
~$
```

QUESTION 23:-WHAT IS USE OF "RMDIR" COMMAND? Removes a directory.

Syntax: -rmdir [-p] directory...

Options:-
-p = Each directory argument is treated as a pathname of which all components will be removed.
-v,--verbose = Display verbose information for every directory processed.

Example: - $rmdir –vp mydir1

OUTPUT

```
~/testDir$ rmdir insideDir/
~/testDir$ cd insideDir
bash: cd: insideDir: No such file or directory
~/testDir$
```

QUESTION 24:-WHAT IS USE OF "LS" COMMAND?

Lists the content of the directory. Syntax: -ls[option]...[FILE]...

Options:-
-I,-inode = Print the index of each file .
-l = Use a long listing format.
-S = Sort by file size.

Example: - $ls –li

OUTPUT

```
~/testDir$ ls
file1.txt  file2.txt  file3.txt
~/testDir$ ls -l
total 2
-rw-r--r-- 1 user user 0 Apr 21 09:54 file1.txt
-rw-r--r-- 1 user user 0 Apr 21 09:54 file2.txt
-rw-r--r-- 1 user user 0 Apr 21 09:54 file3.txt
~/testDir$
```

QUESTION 25:-WHAT IS USE OF "CAT" COMMAND?
Cat stands for catenate. It reads data from files,and output their contents.

Syntax: -cat[option]...[file]...

Options:-
-b,--number-nonblank = Number nonempty output lines ; overrides –n.
-E = Display "$" at end of each line.

Example: - $ cat file1.txt file2.txt

OUTPUT

```
root@kali:~/Documents# cat file.txt

hello world
93547
hi

root@kali:~/Documents# cat -b file.txt

     1  hello world
     2  93547
     3  hi

root@kali:~/Documents# cat -n file.txt
     1
     2  hello world
     3  93547
     4  hi
     5
root@kali:~/Documents# cat -E file.txt
$
hello world $
93547$
hi $
$
```

QUESTION 26:-WHAT IS USE OF “CP” COMMAND?

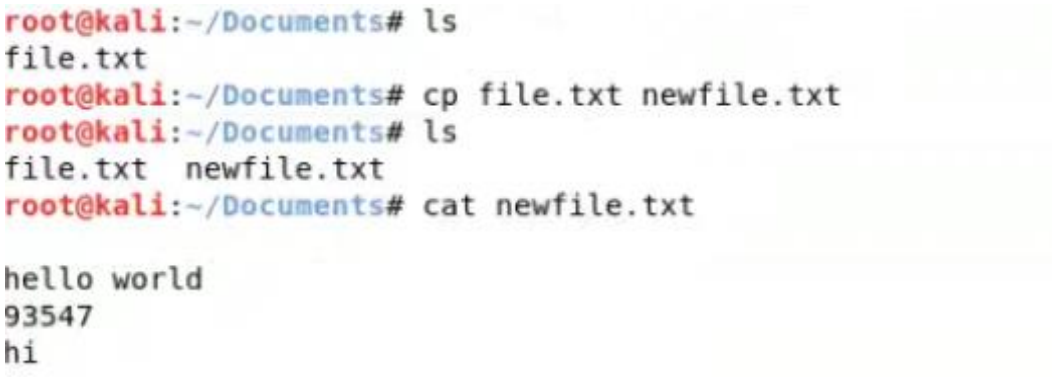
Copies file and directories.

Syntax: -cp [option]... [-T]SOURCE DEST

Options:-
-attributes-only = Don’t copy the file data, just create a file with the same attributes.
--backup = Make a backup of each existing destination file.
-l,--link = Create hardlink to files instead of COPYING them.

Example: - $cp myfile.txt file1.txt

OUTPUT



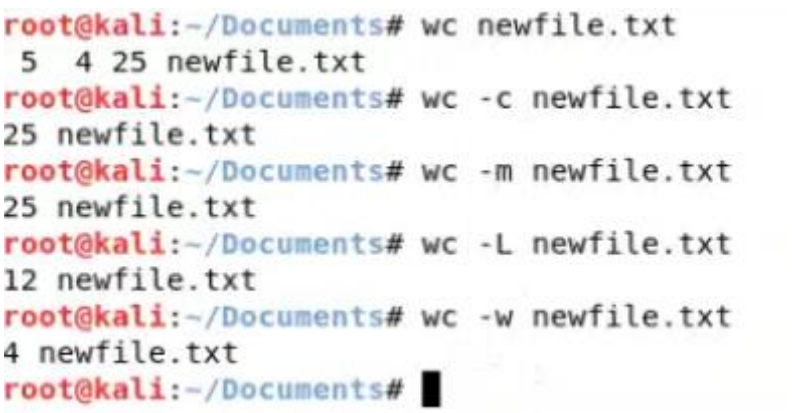
QUESTION 27:-WHAT IS USE OF "WC" COMMAND?
wc or "word count" prints a count of newline words, and bytes for each input file.

Syntax: -wc [OPTION]... [FILE]

Options:-
-c = Print the byte count.
-m = Print the character count.
-l = Print the newline count.
-L = Print the length of longest line.
-w = Print the word counts.

Example: - $wc myfile

OUTPUT



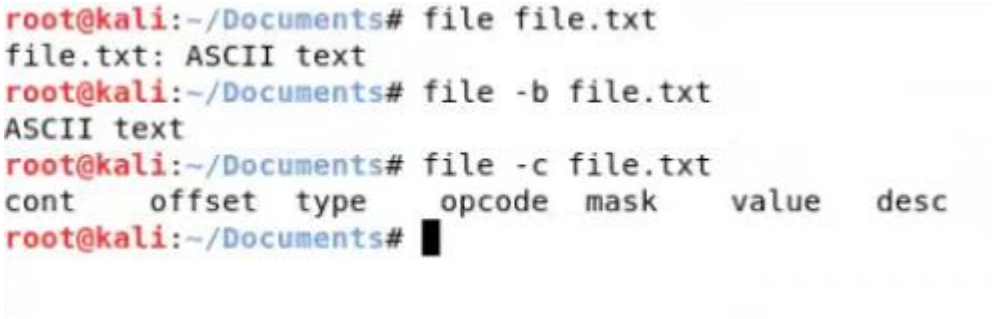
QUESTION 28:-WHAT IS USE OF "FILE" COMMAND? Ans: The file command is used to determine a file's type. Syntax: - file [--help]

Options:-

-b,--brief = Do not prepend file names to output lines.
-c,--compile = write a magic.mgc output filethat contains a pre-parsed version of the magic file or directory.

Example: - file

OUTPUT



QUESTION 29:-WHAT IS USE OF "MV" COMMAND? The mv command is used to move or rename files.

Syntax: - mv [option]... [-t]SOURCE DEST
Options:-
-backup [-CONTROL] = Make a backup of each existing destination file, using the CONTROL. -b = Like –backup, but does not accept an argument, the default backup method is used.

Example: - $mv myfile.txt

OUTPUT

(moving a "file.txt) to "testDir")

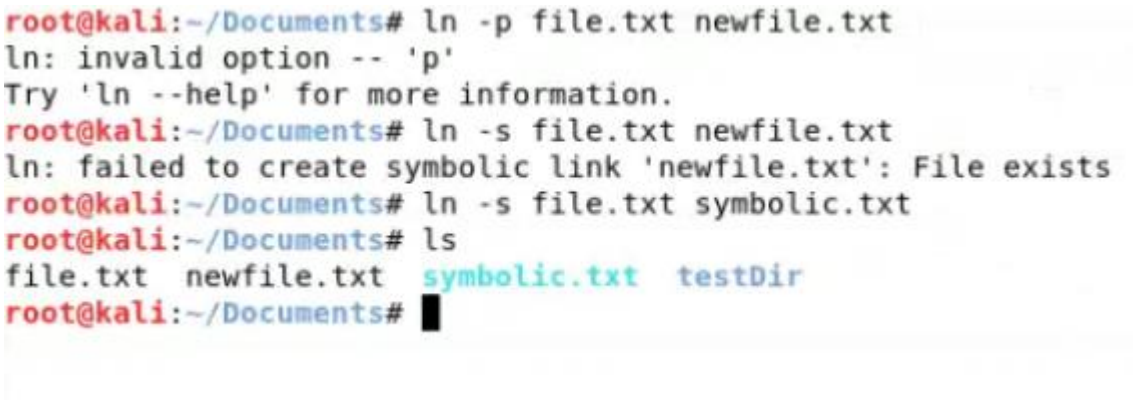
QUESTION 30:- WHAT IS USE OF "LN" COMMAND? This command creates the link between files.

Syntax: -ln[OPTION]...TARGET[...][LINKNAME[...]]

Options:-
-backup = Make a backup of each existing destination file.
-p,--physical = Make hard link directly to symbolic lin.
-s,--symbolic = Make symbolic link instead of hard link.

Example: - $ln –v prmfile.htm prmfile3.htm OUTPUT

OUTPUT



QUESTION 31:-WHAT IS USE OF "CHMOD" COMMAND?
This command is used to change the permission of a file using some special symbols.

Syntax: -chmod[option]...MODE[FILENAME]

Options:-

+:- Used to add permission.
-: - Used to delete permission.
=:- Used to set permission.

Example: - $chmod –v u+x file1.txt

OUTPUT

```
root@kali:~/Documents# ls -li
total 8
553803 -rw-r--r-- 1 root root   32 Jun 11 14:40 file.txt
540152 -rw-r--r-- 1 root root    0 Jun 11 14:44 newfile.txt
540150 lrwxrwxrwx 1 root root    8 Jun 11 14:45 symbolic.txt -> file.txt
553721 drwxr-xr-x 2 root root 4096 Jun 11 14:41 testDir
root@kali:~/Documents# chmod 111 file.txt
root@kali:~/Documents# ls -li
total 8
553803 ---x--x--x 1 root root   32 Jun 11 14:40 file.txt
540152 -rw-r--r-- 1 root root    0 Jun 11 14:44 newfile.txt
540150 lrwxrwxrwx 1 root root    8 Jun 11 14:45 symbolic.txt -> file.txt
553721 drwxr-xr-x 2 root root 4096 Jun 11 14:41 testDir
root@kali:~/Documents#
```

QUESTION 32:- WHAT IS use of “MOUNT” & “UNMOUNT” command?

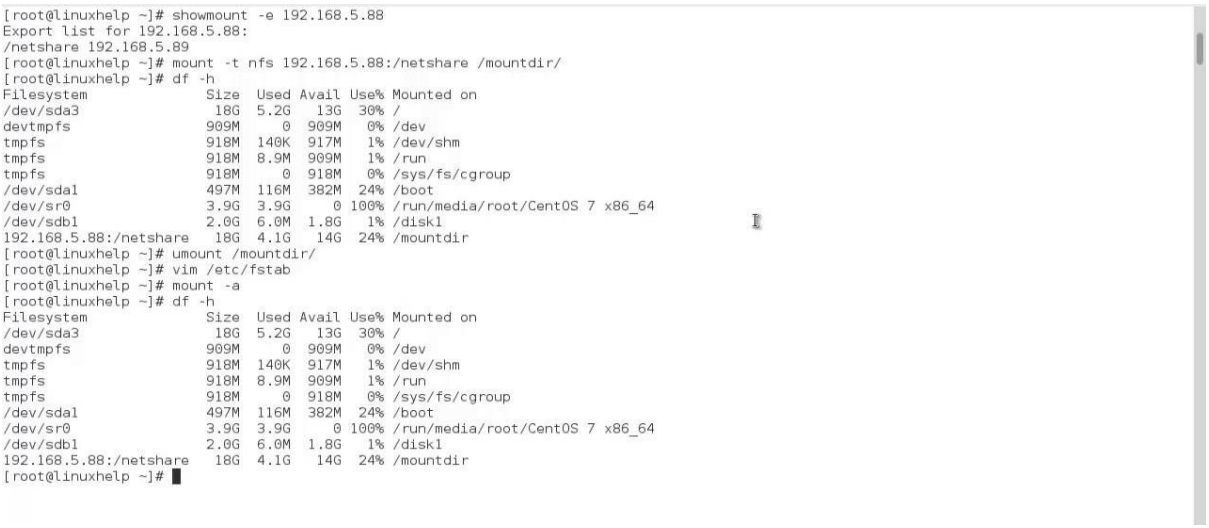
Ans. It helps to mount a file system. This command tells the Kernel to attach file system found at device to the dir , Conversely , another command unmount can be used to detach these device from the file system Tree.

Options:-

-h = prints help message

-V = print a version string

Example:- mount /cd
OUTPUT



QUESTION 33:-WHAT IS USE OF “LESS” COMMAND?
The less program allows you to scroll forward and backward through a text file.

Syntax: - less [FILENAME]
Options:-
-Page Up or b = Scroll back one page.
-Page Down or space = Scroll forward one page. -up Arrow = Scroll up one line.
-Down Arrow = Scroll down one line.

Example: - $less file1.txt

OUTPUT

```
file1
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)
```

QUESTION 34:-WHAT IS USE OF “MORE” COMMAND? Display text, one screen at a time

Syntax: - more [-dlfpcsu][-num lines][+/pattern][+linenum][file...]

Options:-

-p = Do not scroll. Instead, clear the whole screen and then display the text.
-u = Do not display the underline.

Example: - $more +1 file1.txt

OUTPUT

```
file1
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
file1.txt (END)
```

QUESTION 35:-WHAT IS THE USE OF "CUT" COMMAND?

Ans. The cut utility cuts out selected portions of each line (as specified by list) from each _le and writes them to the standard output. If no file arguments are specified, or a file argument is a single dash (`-'), cut reads from the standard input. The items specified by list can be in terms of column position or in terms of fields delimited by a special character. Column numbering starts from 1.

Syntax

cut -c list [file ...]
cut -f list [-d delim] [file ...]

The list is a comma separated list of numbers or number ranges. A number range is of the form a-b where all the columns or fields numbered from a to b (inclusive) are selected

Options

-b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma

-c (column): To cut by character use the -c option. This selects the characters given to the -c option. This can be a list of numbers separated comma or a range

-f (field): -c option is useful for fixed-length lines. Most unix files doesn't have fixed-length lines. To extract the useful information you need to cut by fields rather than columns

OUTPUT

```
┌──(kali㉿kali)-[~]
└─$ cat  file1.txt

 ihey hello
how are u
12324
243vz
vz
cx

┌──(kali㉿kali)-[~]
└─$ cut -c 1-7 file1.txt

 ihey h
how are
12324
243vz
vz
cx

┌──(kali㉿kali)-[~]
└─$ cut -b 1-2 file1.txt

 i
ho
12
24
vz
cx
```

QUESTION 36:- WHAT IS THE USE OF “GZIP”&“GUNZIP” COMMAND?

Ans.These command is used to compress or expand files.

SYNTAX:

1. gzip-[option(if required)] [file_name]
2. gunzip-[option(if required)] [file_name]

OPTIONS:

gzip-[d] (decompress or uncompress.)

gzip -[l] (For each compressed file, list the following fields: compressed size: size of the compressed file uncompressed size: size of the uncompressed file ratio: compression ratio (0.0% if unknown) uncompressed_name: name of the uncompressed file.)

OUTPUT:

```
┌──(kali㉿kali)-[~]
└─$ gzip f.txt

┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  index.html  Pictures  Templates
Desktop  Downloads  f.txt.gz   Music       Public    Videos

┌──(kali㉿kali)-[~]
└─$ gzip -l f.txt
         compressed        uncompressed  ratio uncompressed_name
                 39                  13 -15.4% f.txt

┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  index.html  Pictures  Templates
Desktop  Downloads  f.txt.gz   Music       Public    Videos
```

QUESTION 37:- WHAT IS THE USE OF “BZIP2”&“BUNZIP2” COMMAND?

Ans. These command is used to compress or expand files using aa block-sorting file compressor.

SYNTAX:

1. bzip2 [file_name]
2. bunzip2 [file_name]

OUTPUT:

```
┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  Music     Public     Videos
Desktop  Downloads  f.txt      Pictures  Templates

┌──(kali㉿kali)-[~]
└─$ vim index.html

┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  index.html  Pictures  Templates
Desktop  Downloads  f.txt      Music       Public    Videos

┌──(kali㉿kali)-[~]
└─$ bzip2 index.html

┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  index.html.bz2  Pictures  Templates
Desktop  Downloads  f.txt      Music           Public    Videos

┌──(kali㉿kali)-[~]
└─$ bunzip2 index.html.bz2

┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  index.html  Pictures  Templates
Desktop  Downloads  f.txt      Music       Public    Videos
```

QUESTION 38:- WHAT IS THE USE OF "ZIP"&"UNZIP" COMMAND?

Ans. zip - package and compress (archive) files.
unzip - list, test and extract compressed files in a ZIP archive.

SYNTAX:

1. zip [archive_name] [file1 file2 file3...]

2. unzip [file_name.zip]

OUTPUT:

```
root@kali:~/Documents# ls
file1.txt  file2.txt  file.txt  index.html  newfile.txt  symbolic.txt  testDir
root@kali:~/Documents# zip ab.tar file.txt file1.txt
  adding: file.txt (deflated 3%)
  adding: file1.txt (stored 0%)
root@kali:~/Documents# unzip ab.tar
Archive:  ab.tar
replace file.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: file.txt
replace file1.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: file1.txt
root@kali:~/Documents#
```

QUESTION 39:- WHAT IS THE USE OF “TAR” COMMAND?

Ans. Tar command is the GNU version of the tar archiving utility.

SYNTAX: tar -[option(if required)] [archive_name] [file1 file2 fle3...]

OPTIONS:
tar -[c] = create a new archive.
tar -[x] = (extract files from an archive.)
tar -[f] = (use archive file or device ARCHIVE.)
tar-[t] =(list the contents of an archive.)
tar-[v] =(verbosely list files processed.)

OUTPUT:

```
root@kali:~/Documents# tar -cvf a.tar file.txt file1.txt
file.txt
file1.txt
root@kali:~/Documents# tar -xvf a.tar file.txt file1.txt
file.txt
file1.txt
root@kali:~/Documents# tar -tvf a.tar file.txt file1.txt
---x--x--x root/root        32 2023-06-11 14:40 file.txt
-rw-r--r-- root/root        29 2023-06-11 14:57 file1.txt
root@kali:~/Documents#
```

QUESTION 40:- WHAT IS THE USE OF "PR" COMMAND? Ans. This command is used to convert text files for printing.

Ans: This command is used to convert text files for printing

Syntax: pr [-option] [file name]

Options: pr-[d] (double space the output.)
pr -[t] (omit page headers and trailers.)
pr -[n] (counting starts with 1st line of input file.)
pr -[o] (offset each line with MARGIN (zero) spaces.)
pr – [l] (set the page length to PAGE_LENGTH (66) lines.)

OUTPUT:

```
root@kali:~/Documents# pr +1 file1.txt

2023-06-11 14:57                    file1.txt                    Page 1


hello
hi
why
123
2435757
```

QUESTION 41:- WHAT IS THE USE OF “HEAD” COMMAND?

Ans. This command is used to print the first 10 lines of each FILE to standard output by default.

SYNTAX: head -[option(if required)] [file_name]

OPTIONS: head -[n](print the first K lines instead of the first 10; with the leading '-', Printall but the last K lines of each file.)

OUTPUT:

```
root@kali:~/Documents# head -n 1 file.txt
hello world
root@kali:~/Documents#
```

QUESTION 42:- WHAT IS THE USE OF “TAIL” COMMAND?

Ans. This command is used to print the last 10 lines of each FILE to standard output by default.

SYNTAX: tail -[option(if required)] [file_name]

OPTIONS:

tail -[n](prints last K lines, instead of the last 10; or use -n +K to output starting with the Kth.)

tail -[c] (output the last K bytes; or use -c +K to output bytes starting with Kth of each file.)

OUTPUT:

QUESTION 43:- WHAT IS THE USE OF “PASTE” COMMAND? Ans. This command is used to merge lines of files.

Ans :

SYNTAX: paste -[option(if required)] [file_name]

OPTIONS:

paste -[d] (--delimiters=LIST-reuse characters from LIST instead of TABs.)
paste -[s] (--serial-paste one file at a time instead of in parallel.)

OUTPUT:

```
root@kali:~/Documents# paste file.txt file1.txt
hello world
1234    hello
1233r   hi
asd     why
qwe     123
        2435757

root@kali:~/Documents#
```

QUESTION 44:- WHAT IS THE USE OF "SORT" COMMAND?

Ans. This command is used to sort lines of text files.

SYNTAX: sort -[option(if required)] [file_name]

OPTIONS:

sort -n [--numeric-sort-compare according to string numerical value]

sort -r [--reverse reverse the result of comparisons]

OUTPUT

```
root@kali:~/Documents# sort file.txt

1233r
1234
asd
hello world
qwe
root@kali:~/Documents# sort -n file.txt

asd
hello world
qwe
1233r
1234
root@kali:~/Documents# sort -n -r file.txt
1234
1233r
qwe
hello world
asd

root@kali:~/Documents#
```

QUESTION 45:- WHAT IS THE USE OF “UNIQ” COMMAND?

Ans. This command is used to report or omit repeated lines in a pre-sorted file.

SYNTAX: uniq-[option(if required)] [file_name]

OPTIONS:

uniq -[c] (--count-prefix lines by the number of occurrences.)

uniq –[d] (--repeated-only print duplicate lines, one for each group.)

OUTPUT:

```
root@kali:~/Documents# uniq file.txt
hello world
1234
1233r
asd
qwe

root@kali:~/Documents#
```

QUESTION 46:- WHAT IS THE USE OF “TR” COMMAND?

Ans. This command is used to translate or delete characters.

SYNTAX: tr ['expression1'] ['expression2'] [standard_input(<)] [file_name]

OUTPUT:

```
root@kali:~/Documents# tr he HE <  file.txt
HEllo world
1234
1233r
asd
qwE

root@kali:~/Documents#
```

QUESTION 47:- WHAT IS THE USE OF “CMP” COMMAND?

Ans. This command is used to compare two files byte by byte between pre-sorted files.
SYNTAX: cmp [file_name1] [file_name2]

OUTPUT:

```
┌──(kali㉿kali)-[~]
└─$ cat  f.txt
hey hi
bie

┌──(kali㉿kali)-[~]
└─$ cat  file1.txt

 ihey hello
how are u
12324
243vz
vz
cx

┌──(kali㉿kali)-[~]
└─$ cmp f.txt file1.txt
f.txt file1.txt differ: byte 1, line 1
```

QUESTION 48:- WHAT IS THE USE OF “COMM” COMMAND?

Ans. This command is used to compare two sorted files line by line.

SYNTAX: comm [file_name1] [file_name2]

OUTPUT:

```
┌──(kali㉿kali)-[~]
└─$ comm f.txt file1.txt

hey hi
comm: file 1 is not in sorted order
bie
         ihey hello
comm: file 2 is not in sorted order
        how are u
        12324
        243vz
        vz
        cx
comm: input is not in sorted order
```

QUESTION 49:- WHAT IS THE USE OF “DIFF” COMMAND?

Ans. This command is used to compare files line by line and shows steps to make them identical if they are not.

SYNTAX: diff [file_name1] [file_name2]

OUTPUT:

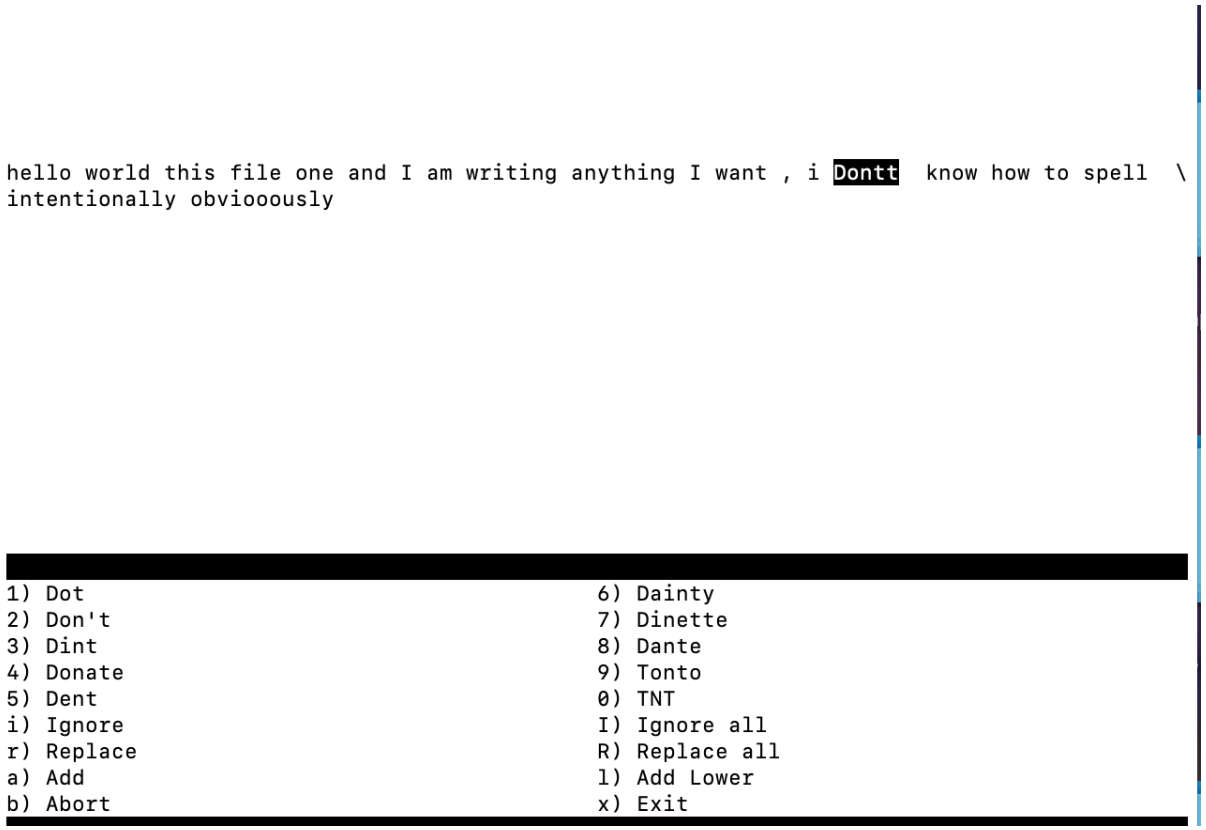
```
┌──(kali㉿kali)-[~]
└─$ ls
cat      Documents  file1.txt  Music     Public     Videos
Desktop  Downloads  f.txt      Pictures  Templates

┌──(kali㉿kali)-[~]
└─$ diff f.txt file1.txt
1,2c1,7
< hey hi
< bie
---
>
>  ihey hello
> how are u
> 12324
> 243vz
> vz
> cx
```

QUESTION 50:- WHAT IS THE USE OF “ASPELL” COMMAND?

Ans. This command is used as interactive spell checker.

SYNTAX: aspell [check] [file_name]

OUTPUT:



QUESTION 51:- WHAT IS THE USE OF "GREP" COMMAND?

Ans. This command is used to print lines matching a pattern.

OPTIONS:

grep [-abcdDEFGHhIiJLlMmnOopqRSsUVvwXxZz] [-A num] [-B num] [-C[num]]
[-e pattern] [-f file] [--binary-files=value] [--color=when]
[--context[=num]] [--directories=action] [--label] [--line-buffered]
[--null] [pattern] [file ...]

OUTPUT:

```
root@kali:~/Documents# cat file.txt
hello world
1234
1233r
asd
qwe

root@kali:~/Documents# grep h file.txt
hello world
root@kali:~/Documents# grep -i file.txt
h
^C
root@kali:~/Documents# grep -i h file.txt
hello world
root@kali:~/Documents# grep -w  h file.txt
root@kali:~/Documents# grep -e  h -e a  file.txt
hello world
asd
root@kali:~/Documents#
```

QUESTION 52:- WHAT IS THE USE OF “SED” COMMAND?

Ans. This command is a stream editor for filtering and transforming text.
SYNTAX: sed-[option(if required)] ['address action(p/q)'] [file_name]
OPTIONS: sed-[n](--quiet, --silent-suppress automatic printing of pattern space.)

OUTPUT:

```
┌──(kali㉿kali)-[~]
└─$ cat > f.txt
hey hi
bie
bye  ^C

┌──(kali㉿kali)-[~]
└─$ sed 's/unix/linux' f.txt
sed: -e expression #1, char 12: unterminated `s' command
```

QUESTION53:-WHAT IS USE OF "HISTORY" COMMAND? The history command list all the previous executed commands.

Syntax:- history [-c] [-d offset] [n]

Options:-
-c=clear the history list by deleting all of the entries.
-d=delete the history entry at OFFSET.
-w=write the current history to the history file and append them to the history list. Syntax :-
$ history

OUTPUT

```
┌──(kali㉿kali)-[~]
└─$ history
    1  cd /Documents
    2  cd Documents/
    3  cd Documents/prgm/
    4  ls
    5  cd Documents/prgm/
    6  cd prgm/
    7  clear
    8  vim 71.sh
    9  chmod +x 71.sh
   10  zsh 71.sh
   11  vim 71.sh
   12  zsh 71.sh
   13  clear
   14  vim 71.sh
   15  chmod +x 71.sh
   16  zsh 71.sh
   17  clear
   18  top
   19  clear
   20  pstree
   21  clear
   22  ls -l
   23  chmod +t file.txt
   24  clear
   25  cat file2.txt
   26  cat >file2.txt
   27  umask
   28  umask 543
   29  clear
   30  tee -a file1.txt
   31  cat file1.txt
   32  touch file.txt
```

QUESTION 54:-WHAT IS THE USE OF TEE COMMAND?

Ans: It read from the standard input and write to standard output and to file

Syntax:-tee [option]...[file]

Option:--a, --append =Append to the given FILES. Do not overwrite.

OUTPUT

```
root@kali:~/Documents# cat file.txt
 hi how are you
1234
why are u here
97765

root@kali:~/Documents# tee -a file.txt
hey :wq
hey :wq


^C
root@kali:~/Documents# cat file.txt
 hi how are you
1234
why are u here
97765

hey :wq

root@kali:~/Documents#
```

QUESTION 55:-WHAT IS THE USE OF UMASK?

Ans : Return or set the value of the system file mode creation mask
Syntax:-umask [–s]... [mask definition]

Option:-
-S= Accept symbolic representation of mask definition, or return one.
Mask Definition = if valid m.d is specified, the unmask is set to this value. Else set to current unmask value.

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ cat >file2.txt
hi how r u ^C

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ umask
022

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ umask 543
```

QUESTION 56:-WHAT IS THE USE OF SUID, SGID, STICKY BIT?

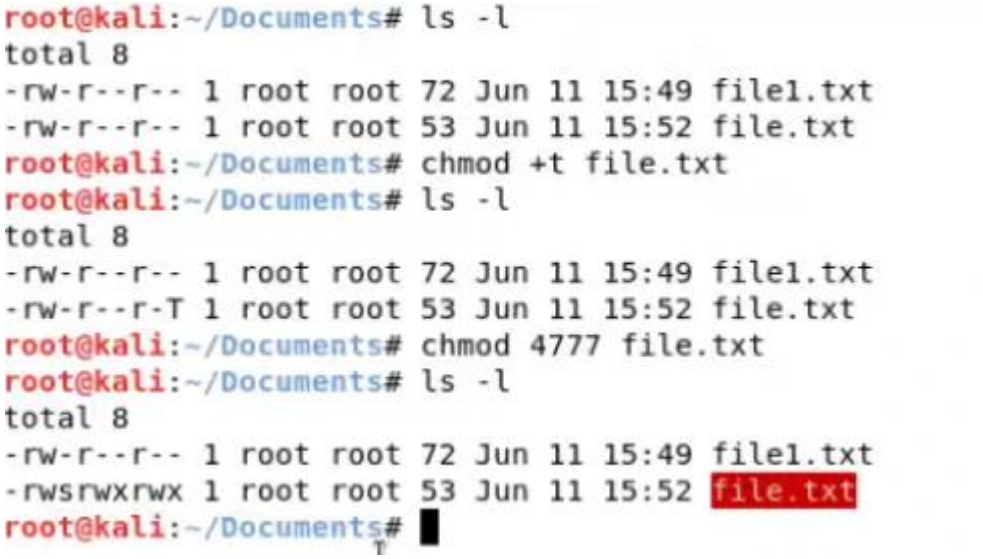
1. SUID (Set User ID): When a file with the SUID bit is executed, it runs with the privileges of the file owner instead of the user who executed it. This is often used for executable files that need elevated privileges to run, such as system utilities like passwd.

2. SGID (Set Group ID): When a file with the SGID bit is executed, it runs with the privileges of the file's group instead of the user who executed it. This is often used for files and directories that need to be shared by a group of users, such as project directories in a development team.

3. Sticky Bit: When a directory has the Sticky Bit set, users can only delete or rename files that they own, regardless of the directory's permissions. This is often used for shared directories like /tmp or /var/tmp, where multiple users need to create and access files, but you don't want users to be able to delete or modify other users' files.

Overall, these permission bits can help improve security and provide finer-grained control over file access and deletion.

OUTPUT



QUESTION-57: What is the use of “ps” command ?

Ans :The "ps" command is used to display information about the currently running processes on a Unix or Linux system. It can show the process ID, CPU and memory usage, status, owner, and other details.

Here are some common options available with the "ps" command:

- "-e": display information about all processes on the system
- "-u": display information about processes owned by a specific user
- "-f": display a full-format listing with more details
- "-aux": a combined option that shows all processes with user, CPU, and memory usage information.

OUTPUT

```
root@kali:~/Documents# ps
  PID TTY          TIME CMD
 2743 pts/0    00:00:00 bash
 3527 pts/0    00:00:00 ps
root@kali:~/Documents#
```

QUESTION-58: What is the use of pstree command?

Ans:

The `pstree` command is a Linux/Unix utility that displays the processes running on the system in the form of a tree. It provides a graphical representation of the processes and their relationships to each other, making it easier to understand how processes are related and to identify any dependencies between them.

Some commonly used options for `pstree` are:

- `-p`: Show the process IDs of each process in the tree.
- `-u`: Show the user who owns each process in the tree.
- `-a`: Show the command line arguments for each process in the tree.
- `-n`: Sort the tree by process ID instead of alphabetically.
- `-h`: Highlight the current process and its ancestors in the tree.
- `-c`: Don't compress the output of the tree, show each branch separately.

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ pstree
systemd─┬─ModemManager───3*[{ModemManager}]
        ├─NetworkManager───3*[{NetworkManager}]
        ├─agetty
        ├─colord───3*[{colord}]
        ├─cron
        ├─dbus-daemon
        ├─haveged
        ├─lightdm─┬─Xorg───{Xorg}
        │         ├─lightdm─┬─xfce4-session─┬─Thunar───3*[{Thunar}]
        │         │         │               ├─agent───3*[{agent}]
        │         │         │               ├─blueman-applet───4*[{blueman-applet}]
        │         │         │               ├─light-locker───4*[{light-locker}]
        │         │         │               ├─nm-applet───4*[{nm-applet}]
        │         │         │               ├─polkit-gnome-au───3*[{polkit-gnome-au}]
        │         │         │               ├─ssh-agent
        │         │         │               ├─xfce4-notifyd───3*[{xfce4-notifyd}]
        │         │         │               ├─xfce4-panel─┬─panel-1-whisker───3*[{panel-1-+
        │         │         │               │            ├─panel-13-cpugra───3*[{panel-13+
        │         │         │               │            ├─panel-14-systra───3*[{panel-14+
        │         │         │               │            ├─panel-15-genmon───3*[{panel-15+
        │         │         │               │            ├─panel-16-pulsea───3*[{panel-16+
        │         │         │               │            ├─panel-17-notifi───3*[{panel-17+
        │         │         │               │            ├─panel-18-power-───3*[{panel-18+
        │         │         │               │            ├─panel-22-action───3*[{panel-22+
        │         │         │               │            └─3*[{xfce4-panel}]
        │         │         │               ├─xfce4-power-man───3*[{xfce4-power-man}]
        │         │         │               ├─xfdesktop───3*[{xfdesktop}]
        │         │         │               ├─xfsettingsd───3*[{xfsettingsd}]
        │         │         │               ├─xfwm4───3*[{xfwm4}]
        │         │         │               ├─xiccd───3*[{xiccd}]
        │         │         │               └─3*[{xfce4-session}]
        │         │         └─3*[{lightdm}]
        │         └─3*[{lightdm}]
        ├─polkitd───3*[{polkitd}]
        ├─qterminal─┬─zsh───zsh───bash───bash───bash
        │           └─6*[{qterminal}]
        ├─qterminal─┬─zsh───pstree
        │           └─2*[{qterminal}]
        ├─rtkit-daemon───2*[{rtkit-daemon}]
        ├─systemd─┬─(sd-pam)
        │         ├─at-spi-bus-laun─┬─dbus-daemon
        │         │                 └─4*[{at-spi-bus-laun}]
        │         ├─at-spi2-registr───3*[{at-spi2-registr}]
        │         ├─dbus-daemon
        │         ├─dconf-service───3*[{dconf-service}]
        │         ├─gnome-keyring-d───4*[{gnome-keyring-d}]
        │         ├─gpg-agent
        │         ├─gvfs-afc-volume───4*[{gvfs-afc-volume}]
        │         ├─gvfs-goa-volume───3*[{gvfs-goa-volume}]
        │         ├─gvfs-gphoto2-vo───3*[{gvfs-gphoto2-vo}]
        │         ├─gvfs-mtp-volume───3*[{gvfs-mtp-volume}]
        │         ├─gvfs-udisks2-vo───4*[{gvfs-udisks2-vo}]
```

QUESTION 59:-WHAT IS THE USE OF NICE COMMAND?

nice execute a utility with an altered scheduling priority

nice runs utility at an altered scheduling priority. If an increment is given, it is used; otherwise an increment of 10 is assumed. The super-user can run utilities with priorities higher than normal by using a negative increment. The priority can be adjusted over a range of -20 (the highest) to 20 (the lowest).

Syntax : nice [-n increment] utility [argument ...]

```
root@kali:~/Documents# nice ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  2743  2737  0  80   0 -  1902 do_wai pts/0    00:00:00 bash
0 R     0  3559  2743  0  90  10 -  2085 -      pts/0    00:00:00 ps
root@kali:~/Documents#
```

QUESTION-60:What is the use of TOP command?

The `top` command is a Linux/Unix utility that provides a real-time, dynamic view of the processes running on the system, including their resource utilization such as CPU, memory, and I/O. It is commonly used to monitor system performance and identify resource-hungry processes.

Some commonly used options for `top` are:

- `-d <seconds>`: Specifies the delay between updates in seconds. By default, `top` updates every 3 seconds.
- `-b`: Runs `top` in batch mode and outputs the result to standard output. This is useful for scripting purposes.
- `-n <iterations>`: Specifies the number of iterations `top` should run before exiting.
- `-p <pid1>,<pid2>,...`: Shows information only for the specified process IDs.
- `-u <username>`: Shows information only for processes owned by the specified user.
- `q`: Quits `top`.

When `top` is running, you can press various keys to interact with it. Some common keys are:

- `k`: Kills a process by entering the process ID.

OUTPUT

```
top - 15:28:04 up  5:40,  1 user,  load average: 0.07, 0.10, 0.09
Tasks: 202 total,   1 running, 201 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  1.5 sy,  0.0 ni, 97.7 id,  0.0 wa,  0.0 hi,  0.4 si,  0.0 st
MiB Mem :   1959.0 total,    304.5 free,    910.4 used,    932.7 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1048.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
863	root	20	0	482136	161060	75576	S	1.0	8.0	2:55.74	Xorg
168081	kali	20	0	11616	5488	3348	R	1.0	0.3	0:00.12	top
1421	kali	20	0	423616	30728	21296	S	0.7	1.5	1:03.15	panel-15-genmon
473	root	20	0	314900	13320	7684	S	0.3	0.7	2:15.23	vmtoolsd
1074	kali	20	0	9736	5564	4332	S	0.3	0.3	0:07.47	dbus-daemon
1196	kali	20	0	471492	49480	33852	S	0.3	2.5	1:22.69	xfwm4
1419	kali	20	0	431952	42396	22288	S	0.3	2.1	1:08.42	panel-13-cpugra
115427	kali	20	0	441388	104620	85704	S	0.3	5.2	0:12.62	qterminal
1	root	20	0	167744	12384	9148	S	0.0	0.6	0:02.49	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_flushwq
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-event+
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kth+
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kt+
14	root	20	0	0	0	0	S	0.0	0.0	0:00.51	ksoftirqd/0
15	root	20	0	0	0	0	I	0.0	0.0	0:10.06	rcu_preempt
16	root	rt	0	0	0	0	S	0.0	0.0	0:00.16	migration/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.19	migration/1
21	root	20	0	0	0	0	S	0.0	0.0	0:00.44	ksoftirqd/1
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-event+
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2

QUESTION 61:-Write a shell script to calculate the area of a triangle.

```
echo "Enter the height and the base of the triangle : "
read b h
area=$(echo "scale=2;(1/2) * $b * $h"|bc)
echo "Area of a triangle is $area"
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 61.sh
Enter the height and the base of the triangle :
6 7
Area of a triangle is 21.00
```

QUESTION 62:-Write a shell script to calculate the area of a rectangle.

```
echo "Enter the length of the rectangle: "
read length

echo "Enter the width of the rectangle: "
read width

area=$(echo "$length * $width" | bc)

echo "The area of the rectangle is: $area"
```

OUTPUT

```
┌──(kali㉿  kali)-[~/Documents/prgm]
└─$ zsh 62.sh
Enter the length of the rectangle:
2
Enter the width of the rectangle:
3
The area of the rectangle is: 6
```

QUESTION 63:-Write a shell script to calculate the area of a square.

```
echo "Enter side ofsquare"
read num
area=$(expr "$num" \* "$num")
echo "The area of square is $area"
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 63.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 63.sh
enter side of square
2
the area of square is 2*2
```

QUESTION 64:-Write a shell script to find the area of a circle.

```
echo "Enter the radius of a circle :"
read r
area=$(echo "scale=2;3.14*($r*$r)" | bc)
echo "Area of circle is$area"
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 64.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 64.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 64.sh
Enter the radius of a circle :
3
Area of circle is28.26
```

QUESTION 65:- Write a shell script to find whether a number entered is even or odd.

```
echo "Enter a number: "
read num

if (( num % 2 == 0 ))
then
  echo "$num is even."
else
  echo "$num is odd."
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 65.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 65.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 65.sh
Enter a number:
10
10 is even.
```

QUESTION 66:-Write a shell script to find the status of the student according to the given constraints:
If marks >=90... Outstanding
If marks>=75and<90...Excellent
If marks>=60and<75...Very Good
If marks>=50and<60...Good
If marks < 50... Poor

```
echo "Enter student's marks: "
read marks

if [ $marks -ge 90 ]
then
   echo "Outstanding"
elif [ $marks -ge 75 ] && [ $marks -lt 90 ]
then
   echo "Excellent"
elif [ $marks -ge 60 ] && [ $marks -lt 75 ]
then
   echo "Very Good"
elif [ $marks -ge 50 ] && [ $marks -lt 60 ]
then
   echo "Good"
else
   echo "Poor"
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 66.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 66.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 66.sh
Enter student's marks:
90
Outstanding
```

QUESTION 67:-Write a shell script to find whether the year is leap or not.

```
echo "Enter a year: "
read year

if (( year % 4 == 0 && year % 100 != 0 )) || (( year % 400 == 0 )); then
   echo "$year is a leap year"
else
   echo "$year is not a leap year"
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 67.sh
Enter a year:
2004
2004 is a leap year

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 67.sh
Enter a year:
2005
2005 is not a leap year
```

QUESTION 68:-Write a shell script to find the largest number among three.

```
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
read -p "Enter the third number: " num3

if [ $num1 -ge $num2 ] && [ $num1 -ge $num3 ]; then
    echo "$num1 is the largest number."
elif [ $num2 -ge $num1 ] && [ $num2 -ge $num3 ]; then
    echo "$num2 is the largest number."
else
    echo "$num3 is the largest number."
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 68.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 68.sh
Enter the first number:
2
Enter the second number:
3
Enter the third number:
5
5 is the largest number.
```

QUESTION 69:-Write a shell script to find the larger of two numbers

```
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2

if [ $num1 -gt $num2 ]; then
   echo "$num1 is the larger number."
else
   echo "$num2 is the larger number."
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 69.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 69.sh
Enter the first number:
2
Enter the second number:
3
3 is the larger number.
```

QUESTION 70:-Write a shell script to convert the temperature Fahrenheit into Celsius.

```
read -p "Enter the temperature in Fahrenheit: " f

# Convert Fahrenheit to Celsius
c=$(echo "scale=2; ($f - 32) * 5 / 9" | bc)

echo "$f degrees Fahrenheit is equal to $c degrees Celsius."
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 70.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 70.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 70.sh
Enter the temperature in Fahrenheit:
100
100 degrees Fahrenheit is equal to 37.77 degrees Celsius.
```

QUESTION 71:-Write a shell script to illustrate case statement by making 12 cases. Each representing a month in a year.

```
read -p "Enter the temperature in Fahrenheit: " f

# Convert Fahrenheit to Celsius
c=$(echo "scale=2; ($f - 32) * 5 / 9" | bc)

echo "$f degrees Fahrenheit is equal to $c degrees Celsius."
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents/prgm]
└─$ vim 71.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ chmod +x 71.sh

┌──(kali㉿kali)-[~/Documents/prgm]
└─$ zsh 71.sh
Enter the month number [1-12]
3
march
```

QUESTION 72:-Write a shell script for a menu driven program using the arithmetic operators

```
echo "Welcome to the arithmetic program!"

while true; do
    # Display the menu
    echo "Please choose an operation:"
    echo "1. Add"
    echo "2. Subtract"
    echo "3. Multiply"
    echo "4. Divide"
    echo "5. Quit"

    # Read the user's choice
    read -p "Enter your choice [1-5]: " choice

    # Perform the chosen operation
    case $choice in
        1)
            read -p "Enter the first number: " num1
            read -p "Enter the second number: " num2
            result=$(echo "$num1 + $num2" | bc)
            echo "The sum of $num1 and $num2 is $result."
            ;;
        2)
            read -p "Enter the first number: " num1
            read -p "Enter the second number: " num2
            result=$(echo "$num1 - $num2" | bc)
            echo "The difference between $num1 and $num2 is $result."
            ;;
        3)
            read -p "Enter the first number: " num1
            read -p "Enter the second number: " num2
            result=$(echo "$num1 * $num2" | bc)
            echo "The product of $num1 and $num2 is $result."
            ;;
        4)
            read -p "Enter the first number: " num1
            read -p "Enter the second number: " num2
            result=$(echo "$num1 / $num2" | bc)
            echo "The quotient of $num1 and $num2 is $result."
            ;;
        5)
```

```
        echo "Exiting the program. Goodbye!"
        exit 0
        ;;
    *)
        echo "Invalid choice. Please enter a number between 1 and 5."
        ;;
  esac

  echo "----------------------"
done
```

OUTPUT

```
└─$ zsh 72.sh
Welcome to the arithmetic program!
Please choose an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Quit
Enter your choice [1-5]:
1
Enter the first number:
2
Enter the second number:
3
The sum of 2 and 3 is 5.
----------------------
Please choose an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Quit
Enter your choice [1-5]:
6
Invalid choice. Please enter a number between 1 and 5.
----------------------
Please choose an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Quit
Enter your choice [1-5]:
```

QUESTION 73:-Write a shell script to illustrate case statement defining a case for each [A-Z], [a-z] and [0-9]

```
read -p "Enter a character: " char

case $char in
   [A-Z])
      echo "$char is an uppercase letter."
      ;;
   [a-z])
      echo "$char is a lowercase letter."
      ;;
   [0-9])
      echo "$char is a digit."
      ;;
   *)
      echo "$char is not a letter or a digit."
      ;;
esac
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 73.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 73.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 73.sh
Enter a character:
:
: is not a letter or a digit.
```

QUESTION 74:- Write a shell script to illustrate the use date , ls , ps , time.

```
echo "Current date and time:"
date
echo "----------------------"

echo "List of files in the current directory:"
ls
echo "----------------------"

echo "List of running processes:"
ps
echo "----------------------"

echo "Execution time for 'ls' command:"
time ls
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 74.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 74.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 74.sh
Current date and time:
Mon Jun 12 02:18:09 PM EDT 2023
----------------------
List of files in the current directory:
72.sh  73.sh  74.sh  abc  file.sh  prgm
----------------------
List of running processes:
    PID TTY          TIME CMD
 115432 pts/2    00:00:02 zsh
 133585 pts/2    00:00:00 zsh
 133596 pts/2    00:00:00 ps
----------------------
Execution time for 'ls' command:
72.sh  73.sh  74.sh  abc  file.sh  prgm
ls  0.00s user 0.00s system 86% cpu 0.001 total
```

QUESTION 75:-Write a shell script to calculate sum of first n natural numbers.

```
read -p "Enter a positive integer: " n

if [[ $n -lt 1 ]]; then
   echo "Error: Input must be a positive integer."
   exit 1
fi

sum=0
for (( i=1; i<=$n; i++ ))
do
   sum=$(( sum + i ))
done

echo "The sum of the first $n natural numbers is $sum."
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 75.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 75.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 75.sh
Enter a positive integer:
1
The sum of the first 1 natural numbers is 1.
```

QUESTION 76:-Write a shell script to sort n numbers.

```
read -p "Enter the number of elements to be sorted: " n

if [[ $n -lt 1 ]]; then
   echo "Error: Input must be a positive integer."
   exit 1
fi

echo "Enter $n elements:"
read -a arr

# Sort the array using the 'sort' command
sorted_arr=($(echo "${arr[@]}" | tr ' ' '\n' | sort -n))

echo "The sorted array is: ${sorted_arr[@]}"
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 76.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 76.sh
Enter the number of elements to be sorted:
5
Enter 5 elements:
1
The sorted array is: 1

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 76.sh
Enter the number of elements to be sorted:
5
Enter 5 elements:
1 2 9 5 7
The sorted array is: 1 2 5 7 9
```

QUESTION 78.Write a shell script to determine whether the given number is prime or not

```
read -p "Enter a positive integer: " n

if [[ $n -lt 2 ]]; then
    echo "Error: Input must be greater than or equal to 2."
    exit 1
fi

# Check if the number is prime
is_prime=true
for (( i=2; i<$n; i++ ))
do
    if [[ $(( n % i )) -eq 0 ]]; then
        is_prime=false
        break
    fi
done

if [[ $is_prime == true ]]; then
    echo "$n is a prime number."
else
    echo "$n is not a prime number."
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 78.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 78.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 78.sh
Enter a positive integer:
3
3 is a prime number.
```

QUESTION 79:-Write a shell script to find the factorial of a given number.

```
read -p "Enter a positive integer: " n

if [[ $n -lt 0 ]]; then
   echo "Error: Input must be a non-negative integer."
   exit 1
fi

factorial=1

for (( i=1; i<=n; i++ ))
do
   factorial=$(( factorial * i ))
done

echo "The factorial of $n is $factorial."
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 79.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 79.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 79.sh
Enter a positive integer:
4
The factorial of 4 is 24.
```

QUESTION 80:-Write a shell script to print the Fibonacci series

```
read -p "Enter the number of terms in the Fibonacci series: " n

if [[ $n -lt 1 ]]; then
    echo "Error: Input must be a positive integer."
    exit 1
fi

# First two terms of the series
a=0
b=1

echo "The Fibonacci series of $n terms is:"

# Loop to generate subsequent terms
for (( i=0; i<n; i++ ))
do
    echo -n "$a "

    # Calculate the next term
    next=$(( a + b ))

    # Shift the values to generate the next term
    a=$b
    b=$next
done

echo ""
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ zsh 80.sh
Enter the number of terms in the Fibonacci series:
8
The Fibonacci series of 8 terms is:
0 1 1 2 3 5 8 13

┌──(kali㉿kali)-[~/Documents]
└─$
```

QUESTION 81:-Write a shell script to display multiplication table of a number.

```
read -p "Enter a positive integer: " n

if [[ $n -lt 1 ]]; then
   echo "Error: Input must be a positive integer."
   exit 1
fi

echo "Multiplication table for $n:"

# Loop to generate multiplication table
for (( i=1; i<=10; i++ ))
do
   echo "$n x $i = $(( n * i ))"
done
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ zsh 80.sh
Enter the number of terms in the Fibonacci series:
8
The Fibonacci series of 8 terms is:
0 1 1 2 3 5 8 13

┌──(kali㉿kali)-[~/Documents]
└─$ vim 81.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 81.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 81.sh
Enter a positive integer:
25
Multiplication table for 25:
25 x 1 = 25
25 x 2 = 50
25 x 3 = 75
25 x 4 = 100
25 x 5 = 125
25 x 6 = 150
25 x 7 = 175
25 x 8 = 200
25 x 9 = 225
25 x 10 = 250
```

QUESTION 82:-Write a shell script to find the position of a word from the string.

```
# Define the string and the word to find
echo "Enter a string: "
read string
echo "Enter a word to find: "
read word

# Use the grep command to find the word in the string and get its position
position=$(echo $string | grep -b -o $word | awk -F: '{print $1}')

# Check if the word was found in the string
if [ -z "$position" ]; then
  echo "The word \"$word\" was not found in the string \"$string\""
else
  echo "The word \"$word\" was found at position $position in the string \"$string\""
fi
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 82.sh

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x 82.sh

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 82.sh
Enter a string:
hello world unix , i want to know more about unix
Enter a word to find:
want
The word "want" was found at position 21 in the string "hello world unix , i want to know m
ore about unix"
```

QUESTION 83. Write a shell script to count no. of words, characters and blank spaces.

```
# Define the file name
echo "Enter the file name: "
read file

# Count the number of words in the file
word_count=$(cat $file | wc -w)

# Count the number of characters in the file
char_count=$(cat $file | wc -m)

# Count the number of blank spaces in the file
space_count=$(cat $file | tr -cd ' ' | wc -c)

# Print the results
echo "Word count: $word_count"
echo "Character count: $char_count"
echo "Space count: $space_count"
```

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ vim 83.sh

┌──(kali㉿kali)-[~/Documents]
└─$ cat file1.txt
cat: file1.txt: No such file or directory

┌──(kali㉿kali)-[~/Documents]
└─$ vim file1.txt

┌──(kali㉿kali)-[~/Documents]
└─$ zsh 83.sh
Enter the file name:
file1.txt
Word count: 15
Character count: 74
Space count: 14

┌──(kali㉿kali)-[~/Documents]
└─$
```

QUESTION 84. What is the use of Redirection symbol.

Ans : Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices.

Syntax: - command < [input file] > [output file]
Example: - $ bc< file1.txt > file2.txt

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ touch calc.txt

┌──(kali㉿kali)-[~/Documents]
└─$ vi calc.txt

┌──(kali㉿kali)-[~/Documents]
└─$ cat > cal.txt
12-4
12-9
^C

┌──(kali㉿kali)-[~/Documents]
└─$ [SIGINT]> touch result.txt
[SIGINT]: command not found

┌──(kali㉿kali)-[~/Documents]
└─$ touch result.txt

┌──(kali㉿kali)-[~/Documents]
└─$ cat result.txt
```

QUESTION 85. What is the use of Standard Input, Standard Output and Standard Error?

Ans : Standard Input is the keyboard, abstracted as a file to make it easier to write shell scripts; Standard Output is the shell window or the terminal from which the script runs, abstracted as a file to again make writing scripts & program easier; and Standard Error is the same as standard output:the shell window or terminal from which the script runs, storing the errors encountered in a file.
File descriptor 0 refers to the standard input, file descriptor 1 refers to standard output and file descriptor 2 refers to standard error.

Syntax: - command < [input source]
command> [output destination]
command 2> [error destination]

Example: -
$ ls > list.txt $ wc< list.txt
$ wcnofile 2>errorfile

OUTPUT

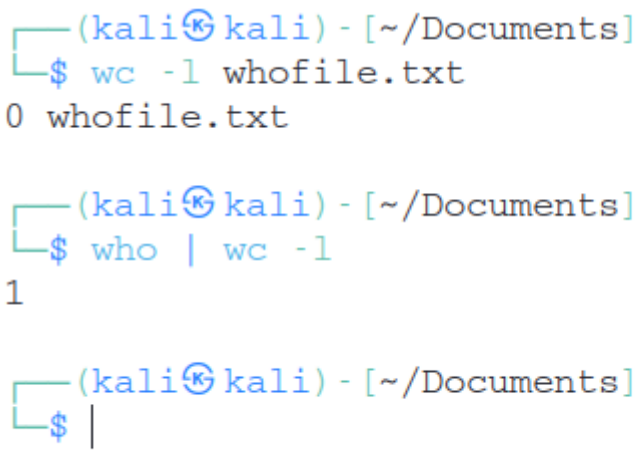
```
┌──(kali㉿kali)-[~/Documents]
└─$ who
kali     tty7         2023-06-12 09:47 (:0)

┌──(kali㉿kali)-[~/Documents]
└─$ who > whofile.txt

┌──(kali㉿kali)-[~/Documents]
└─$ cat whofile.txt
kali     tty7         2023-06-12 09:47 (:0)

┌──(kali㉿kali)-[~/Documents]
└─$ wc -l < whofile.txt
1

┌──(kali㉿kali)-[~/Documents]
└─$ touch nxtwhofile.txt > whofile.txt

┌──(kali㉿kali)-[~/Documents]
└─$ cat whofile.txt > nxtwhofile

┌──(kali㉿kali)-[~/Documents]
└─$ cat nxtwhofile
```

QUESTION 86. What is the use of pipe operator?

Ans :The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. The output of each process is used directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.

Syntax: - $ command_1 | command_2 Example: - $ sort record.txt | uniq

OUTPUT



QUESTION 87. What is the use of VI EDITOR?

Ans:
The VI editor is the most popular and classic text editor in the Linux family. Syntax: - vi ... [SOURCE FILE]
Keystrokes: -
k= Move cursor Up
j = Move cursor Down
h = Move cursor Left
l = Move cursor Right
i = Enter input mode
Esc = Exit input mode
A = Append at end of line r = Replace character

Example: - $vi file.txt

OUTPUT

```
┌──(kali㉿kali)-[~/Documents]
└─$ wc -l whofile.txt
0 whofile.txt

┌──(kali㉿kali)-[~/Documents]
└─$ who | wc -l
1

┌──(kali㉿kali)-[~/Documents]
└─$ vi 87.txt

┌──(kali㉿kali)-[~/Documents]
└─$ cat 87.txt
hello welcome to tha unix world
how are u
```

QUESTION 88. What is the use of “SU” command?

Ans.:- The su command is used to change the current user ID to that of the superuser, or another user.

Syntax:-$ su[options] [username]
Options:-
-c = Specify a command that will be invoked by the shell using its -c.
-s = Invokes the shell
-m = Preserves the current environment

Example:-su – user2

OUTPUT