



MAHARAJA SURAJMAL INSTITUTE

(Affiliated to Guru Gobind Singh Indraprastha University)

LINUX PRACTICAL FILE

Submitted By:

Ayush Negi

02214902020

BCA – 6th Sem (B)

Submitted to:

Prof. Ravinder Kajal

INDEX

S.No	Title	Sign
1.	What is use of "help" command?	
2.	What is the use of "man" command?	
3.	What is the use of "info" command?	
4.	What is the use of "whatis" command?	
5.	What is use of "type" command?	
6.	What is the use of "w" command?	
7.	What is the use of "date" command?	
8.	Bc command	
9.	What is use of "cal" command?	
10.	What is use of "who" command? What is use of "who -hu" command?	
11.	What is use of "whoami" command?	
12.	What is the use of "echo" command?	
13.	What is use of "printf" command?	
14.	What is the use of "tty" command?	
15.	What is use of "stty" 'command?	
16.	What is the use of "path variable"?	
17.	What is use of environment variables?	
18.	What is use of "uname" command?	
19.	What is the use of "hostname" command?	
20.	What is use of "pwd" command?	
21.	What is use of "cd" command?	
22.	What is use of "mkdir" command?	
23.	What is use of "rmdir" command?	
24.	What is use of "ls" command?	
25.	What is use of "cat" command?	
26.	What is use of "cp" command?	
27.	What is use of "wc" command?	
28.	What is use of "file" command?	
29.	What is use of "mv" command?	
30.	What is use of "ln" command?	
31.	What is use of "chmod" command?	
32.	What is use of "mount" and "unmount" command?	
33.	What is use of "less" command?	
34.	What is use of "more" command?	
35.	What is use of "cut" command?	
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, ls, 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?	
89.	What is "awk" ?	
90.	What is "at" 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 pseudomanpage format.

-s = output only a short usage synopsis for each topic matching.

OUTPUT

```
ayush@negiPC28:~$ help
GNU bash, version 5.1.16(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u >
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...] COMMANDS ;;)... esa>
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abdefgjkstuv] [-o option] [-A action] [-G globp>
complete [-abdefgjkstuv] [-pr] [-DEI] [-o option] [-A act>
comptop [-o|+o option] [-DEI] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgiIlNrtux] [-p] [name[=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid ...]
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [name ...]
eval [arg ...]
exec [-cl] [-a name] [command [argument ...]] [redirection>
exit [n]
export [-fn] [name[=value] ...] or export -p
false
history [-c] [-d offset] [n] or history -anrw [filename]>
if COMMANDS; then COMMANDS; [ elif COMMANDS; then COMMAND>
jobs [-lnprs] [jobspec ...] or jobs -x command [args]
kill [-s sigspec | -n signum | -sigspec] pid | jobspec .>
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-d delim] [-n count] [-O origin] [-s count] [-t>
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LP]
read [-ers] [-a array] [-d delim] [-i text] [-n nchars] >
readarray [-d delim] [-n count] [-O origin] [-s count] [>
readonly [-aAf] [name[=value] ...] or readonly -p
return [n]
select NAME [in WORDS ... ;] do COMMANDS; done
set [-abefhkmnptuvxBCHP] [-o option-name] [--] [arg ...]
shift [n]
shopt [-pqsu] [-o] [optname ...]
source filename [arguments]
suspend [-f]
test [expr]
time [-p] pipeline
times
trap [-lp] [[arg] signal_spec ...]
true
type [-afptP] name [name ...]
typeset [-aAfFgiIlNrtux] [-p] name[=value] ...
ulimit [-SHabdefiklmnpqrstuvxPT] [limit]
```

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)                                User Commands                                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 version described here.
    Please refer to your shell's documentation for details about the options it supports.

AUTHOR
    Written by Jim Meyering.

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

COPYRIGHT
    Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
    <https://gnu.org/licenses/gpl.html>.
    This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent per-
    mitted by law.

Manual page pwd(1) line 1 (press h for help or q to quit)
```

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 -h)

```
ayush@neglPC28:~$ info -h
Usage: info [OPTION]... [MENU-ITEM...]

Read documentation in Info format.

Frequently-used options:
  -a, --all                use all matching manuals
  -k, --apropos=STRING      look up STRING in all indices of all manuals
  -d, --directory=DIR       add DIR to INFOPATH
  -f, --file=MANUAL         specify Info manual to visit
  -h, --help               display this help and exit
  --index-search=STRING    go to node pointed by index entry STRING
  -n, --node=NODENAME       specify nodes in first visited Info file
  -o, --output=FILE         output selected nodes to FILE
  -O, --show-options, --usage go to command-line options node
  --subnodes               recursively output menu items
  -v, --variable VAR=VALUE  assign VALUE to Info variable VAR
  --version                display version information and exit
  -w, --where, --location   print physical location of Info file

The first non-option argument, if present, is the menu entry to start from;
it is searched for in all 'dir' files along INFOPATH.
If it is not present, info merges all 'dir' files and shows the result.
Any remaining arguments are treated as the names of menu
items relative to the initial node visited.

For a summary of key bindings, type H within Info.

Examples:
  info                show top-level dir menu
  info info-stand     show the manual for this Info program
  info emacs          start at emacs node from top-level dir
  info emacs buffers  select buffers menu entry in emacs manual
  info emacs -n Files start at Files node within emacs manual
  info '(emacs)Files' alternative way to start at Files node
  info --show-options emacs start at node with emacs' command line options
  info --subnodes -o out.txt emacs dump entire emacs manual to out.txt
  info -f ./foo.info  show file ./foo.info, not searching dir
```

(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)

```
ayush@negiPC28:~$ 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

```
ayush@negiPC28:~$ 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

```
ayush@negiPC28:~$ type mkdir
mkdir is hashed (/usr/bin/mkdir)
ayush@negiPC28:~$ type -a mkdir
mkdir is /usr/bin/mkdir
mkdir is /bin/mkdir
ayush@negiPC28:~$ |
```

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

```
ayush@negiPC28:~$ w
 11:52:46 up 32 min,  0 users,   load average: 0.00, 0.00, 0.00
USER      TTY      FROM              LOGIN@   IDLE   JCPU   PCPU   WHAT
ayush@negiPC28:~$ w -h
ayush@negiPC28:~$ w -h
ayush@negiPC28:~$ w -o
 11:53:35 up 33 min,  0 users,   load average: 0.00, 0.00, 0.00
USER      TTY      FROM              LOGIN@   IDLE   JCPU   PCPU   WHAT
ayush@negiPC28:~$ w -u
 11:54:09 up 34 min,  0 users,   load average: 0.00, 0.00, 0.00
USER      TTY      FROM              LOGIN@   IDLE   JCPU   PCPU   WHAT
ayush@negiPC28:~$ |
```

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

```
ayush@negiPC28:~$ date
Mon Jun  5 11:55:53 IST 2023
ayush@negiPC28:~$ date +%z
+0530
ayush@negiPC28:~$ date +%x
06/05/23
ayush@negiPC28:~$ |
```

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

```
ayush@negiPC28:~$ echo "7+3" | bc
10
ayush@negiPC28:~$ echo "var=20;var" | bc
20
ayush@negiPC28:~$ |
```

QUESTION 9:- WHAT IS USE OF “CAL” COMMAND?

Display a conventionally - 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

(For a particular year)

```
onworks@onworks-Standard-PC-l440FX-PIIX-1996: ~  
onworks@onworks-Standard-PC-l440FX-PIIX-1996:~$ cal -y 2024  
2024  
  Januar          Februar          März  
So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa  
  1  2  3  4  5  6      1  2  3      1  2  
  7  8  9 10 11 12 13    4  5  6  7  8  9 10    3  4  5  6  7  8  9  
14 15 16 17 18 19 20    11 12 13 14 15 16 17    10 11 12 13 14 15 16  
21 22 23 24 25 26 27    18 19 20 21 22 23 24    17 18 19 20 21 22 23  
28 29 30 31            25 26 27 28 29            24 25 26 27 28 29 30  
                               31  
  
  April          Mai          Juni  
So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa  
  1  2  3  4  5  6      1  2  3  4      1  
  7  8  9 10 11 12 13    5  6  7  8  9 10 11    2  3  4  5  6  7  8  
14 15 16 17 18 19 20    12 13 14 15 16 17 18    9 10 11 12 13 14 15  
21 22 23 24 25 26 27    19 20 21 22 23 24 25    16 17 18 19 20 21 22  
28 29 30                26 27 28 29 30 31        23 24 25 26 27 28 29  
                               30  
  
  Juli          August          September  
So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa  
  1  2  3  4  5  6      1  2  3      1  2  3  4  5  6  7  
  7  8  9 10 11 12 13    4  5  6  7  8  9 10    8  9 10 11 12 13 14  
14 15 16 17 18 19 20    11 12 13 14 15 16 17    15 16 17 18 19 20 21  
21 22 23 24 25 26 27    18 19 20 21 22 23 24    22 23 24 25 26 27 28  
28 29 30 31            25 26 27 28 29 30 31    29 30  
  
  Oktober          November          Dezember  
So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa So Mo Di Mi Do Fr Sa  
  1  2  3  4  5      1  2      1  2  3  4  5  6  7  
  6  7  8  9 10 11 12    3  4  5  6  7  8  9    8  9 10 11 12 13 14  
13 14 15 16 17 18 19    10 11 12 13 14 15 16    15 16 17 18 19 20 21  
20 21 22 23 24 25 26    17 18 19 20 21 22 23    22 23 24 25 26 27 28  
27 28 29 30 31        24 25 26 27 28 29 30    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

```
ayush@negiPC28:~$ who
ayush pts/2 2023-06-05 12:17
ayush pts/2 2023-06-05 12:17
ayush@negiPC28:~$ who -Hu
NAME LINE TIME IDLE PID COMMENT
ayush pts/2 2023-06-05 12:17 . 656
ayush pts/2 2023-06-05 12:17 . 730
ayush@negiPC28:~$ |
```

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

```
ayush@negiPC28:~$ whoami
ayush
ayush@negiPC28:~$ whoami --help
Usage: whoami [OPTION]...
Print the user name associated with the current effective user ID.
Same as id -un.

    --help      display this help and exit
    --version   output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>
Full documentation <https://www.gnu.org/software/coreutils/whoami>
or available locally via: info '(coreutils) whoami invocation'
ayush@negiPC28:~$ whoami --version
whoami (GNU coreutils) 8.32
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Richard Mlynarik.
```


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

```
ayush@negiPC28:~$ echo -e "Hello \n World" after quotes
Hello
World after quotes
ayush@negiPC28:~$ echo Hello World
Hello World
ayush@negiPC28:~$ echo -e "Hello \n World \tSee you Later"
Hello
World  See you Later
```

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 expressed in the output.

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

Example: -printf “MAHARAJA SURAJMAL INSTITUTE”

OUTPUT

```
ayush@negiPC28:~$ printf "MAHARAJA SURAJMAL INSTITUTE"
MAHARAJA SURAJMAL INSTITUTEayush@negiPC28:~$ |
```

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

```
ayush@negiPC28:~$ tty  
/dev/pts/2
```

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

```
ayush@negiPC28:~$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
```

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

Example: - echo \$PATH

OUTPUT

```
ayush@negIPC28:/mnt/c$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
ayush@negIPC28:/mnt/c$ |
```

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

OUTPUT

```
ayush@negiPC28:~$ echo $USER
ayush
ayush@negiPC28:~$ echo $SHELL
/bin/bash
ayush@negiPC28:~$ echo $LANG
C.UTF-8
ayush@negiPC28:~$ echo $DISPLAY

ayush@negiPC28:~$ echo $HOME
/home/ayush
ayush@negiPC28:~$ echo $TERM
xterm-256color
ayush@negiPC28:~$ export $LANG=en_US.UTF-8
-bash: export: `C.UTF-8=en_US.UTF-8': not a valid identifier
ayush@negiPC28:~$ export LANG=en_US.UTF-8
ayush@negiPC28:~$ echo $LANG
en_US.UTF-8
ayush@negiPC28:~$ echo $MAIL

ayush@negiPC28:~$
```

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,--kernel-name = Print the kernel name.

-n,--nodename = Print the network mode name.

Example: - \$uname -a

OUTPUT

```
ayush@negiPC28:~$ uname
Linux
ayush@negiPC28:~$ uname -n
negiPC28
ayush@negiPC28:~$ uname -a
Linux negiPC28 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
ayush@negiPC28:~$ uname -s
Linux
ayush@negiPC28:~$ |
```

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

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

Options:-

Example: - \$ hostname

OUTPUT

```
ayush@negiPC28:~$ hostname
negiPC28
ayush@negiPC28:~$ |
```


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

```
ayush@negiPC28:~$ pwd
/home/
ayush@negiPC28:~$ pwd -L
/home/
ayush@negiPC28:~$ pwd -p
-bash: pwd: -p: invalid option
pwd: usage: pwd [-LP]
ayush@negiPC28:~$ pwd -P
/home/
ayush@negiPC28:~$ pwd --help
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.
```

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

```
ayush@negiPC28:~$ cd /mnt/c  
ayush@negiPC28:/mnt/c$ |
```

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

```
ayush@negiPC28:/mnt/c/Users$ cd /mnt/d
ayush@negiPC28:/mnt/d$ mkdir testfile
ayush@negiPC28:/mnt/d$ cd testfile
ayush@negiPC28:/mnt/d/testfile$ cd ..
ayush@negiPC28:/mnt/d$ cd testfile
ayush@negiPC28:/mnt/d/testfile$ mkdir secfile
ayush@negiPC28:/mnt/d/testfile$ cd secfile
ayush@negiPC28:/mnt/d/testfile/secfile$ cd $HOME
ayush@negiPC28:~$ |
```

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 secfile`

OUTPUT

```
ayush@negiPC28:/mnt/d$ cd testfile
ayush@negiPC28:/mnt/d/testfile$ ls secfile
ayush@negiPC28:/mnt/d/testfile$ rmdir secfile
ayush@negiPC28:/mnt/d/testfile$ ls
ayush@negiPC28:/mnt/d/testfile$ |
```

QUESTION 24:-WHAT IS USE OF “LS” COMMAND?

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

Options:-

-l,inode = Print the index of each file .

-l = Use a long listing format.

-S = Sort by file size.

Example: - \$ls -l

OUTPUT

```
ayush@negiPC28:/mnt/d/testfile$ cd ..
ayush@negiPC28:/mnt/d$ ls
'RECYCLE.BIN'      'Downloads 2022'  'Python'          'testfile'
'CLASSIFICATION OF COMPUTERS.pptx'  'NISC'           'System Volume Information'
ayush@negiPC28:/mnt/d$ ls -l
ls: 'System Volume Information': Permission denied
total 740
drwxrwxrwx 1 ayush ayush      512 Jul 11  2022 'RECYCLE.BIN'
-rwxrwxrwx 1 ayush ayush  755044 Aug  9  2022 'CLASSIFICATION OF COMPUTERS.pptx'
drwxrwxrwx 1 ayush ayush      512 Jan 15 19:45 'Downloads 2022'
drwxrwxrwx 1 ayush ayush      512 Jun  4 09:50 'NISC'
drwxrwxrwx 1 ayush ayush      512 Jul 15  2022 'Python'
d--x--x--x 1 ayush ayush      512 Feb 24 23:24 'System Volume Information'
drwxrwxrwx 1 ayush ayush      512 Jun  7 21:07 'testfile'
```

QUESTION 25:-WHAT IS USE OF “CAT” COMMAND?

Cat stands for concatenate. 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 file.txt file1.txt file2.txt

OUTPUT

```
ayush@negiPC28:/mnt/d$ cat file.txt file1.txt file2.txt
filefile1file2ayush@negiPC28:/mnt/d$ |
```

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 file2.txt file3.txt

OUTPUT

```
ayush@negiPC28:/mnt/d$ ls
'$RECYCLE.BIN'      MISC                a.tar      file2.txt    testfile
'CLASSIFICATION OF COMPUTERS.pptx' Pgdm               ab.tar     file3.txt
'Downloads 2022'   'System Volume Information' file.txt    newfie.txt
```

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

```
ayush@negiPC28:/mnt/d$ wc file1.txt
0 1 5 file1.txt
ayush@negiPC28:/mnt/d$ wc -l file1.txt
0 file1.txt
ayush@negiPC28:/mnt/d$ wc -w file1.txt
1 file1.txt
ayush@negiPC28:/mnt/d$ wc -c file1.txt
5 file1.txt
ayush@negiPC28:/mnt/d$ |
```


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 file that contains a pre-parsed version of the magic file or directory.

Example: - file

OUTPUT

```
ayush@negiPC28:/mnt/d$ file file1.txt
file1.txt: ASCII text, with no line terminators
ayush@negiPC28:/mnt/d$ |
```

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 “file1.txt” to “testfile”)

```
ayush@negiPC28:/mnt/d$ ls
'CLASSIFICATION OF COMPUTERS.pptx'  'Downloads 2022'  'System Volume Information'  file  file2.txt  testfile
ayush@negiPC28:/mnt/d$ mv file.txt testfile
mv: cannot stat 'file.txt': No such file or directory
ayush@negiPC28:/mnt/d$ mv file1.txt testfile
ayush@negiPC28:/mnt/d$ cd testfile
ayush@negiPC28:/mnt/d/testfile$ ls
file1.txt
ayush@negiPC28:/mnt/d/testfile$ |
```

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

```
ayush@negiPC28:/mnt/d$ ln -s file2.txt testfile
ayush@negiPC28:/mnt/d$ cd testfile
ayush@negiPC28:/mnt/d/testfile$ ls
file1.txt  file2.txt
ayush@negiPC28:/mnt/d/testfile$ |
```

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

UTPUT

```
ayush@negiPC28:/mnt/d/testfile$ chmod 777 file1.txt
ayush@negiPC28:/mnt/d/testfile$ ls -li
total 0
12384898975271894 -rwxrwxrwx 1          5 Jun  7 21:31 file1.txt
281474976713691 lrwxrwxrwx 1          9 Jun  8 2023 file2.txt -> file2.txt
ayush@negiPC28:/mnt/d/testfile$ cd ..
ayush@negiPC28:/mnt/d$ chmod g+x file.txt
chmod: cannot access 'file.txt': No such file or directory
ayush@negiPC28:/mnt/d$ chmod g+x file.txt
ayush@negiPC28:/mnt/d$ ls -li
ls: 'System Volume Information': Permission denied
total 740
281474976710743 drwxrwxrwx 1 ayush ayush      512 Jul 11 2022 RECYCLE.BIN
844424930132020 -rwxrwxrwx 1 ayush ayush    755044 Aug  9 2022 'CLASSIFICATION OF COMPUTERS.pptx'
1970324836974652 drwxrwxrwx 1 ayush ayush      512 Jan 15 19:45 Downloads 2022
8162774324609077 drwxrwxrwx 1 ayush ayush      512 Jun  4 09:50 misc
844424930132015 drwxrwxrwx 1 ayush ayush      512 Jul 15 2022 Pyton
281474976710694 d--x--x--x 1 ayush ayush      512 Feb 24 23:24 'System Volume Information'
562949953424348 -rwxrwxrwx 1 ayush ayush       4 Jun  8 2023 file.txt
844424930135000 -rwxrwxrwx 1 ayush ayush       5 Jun  7 21:31 file2.txt
281474976713690 -rwxrwxrwx 1 ayush ayush       5 Jun  8 2023 newfie.txt
4785074604082671 drwxrwxrwx 1 ayush ayush      512 Jun  8 2023 testfile
```

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.

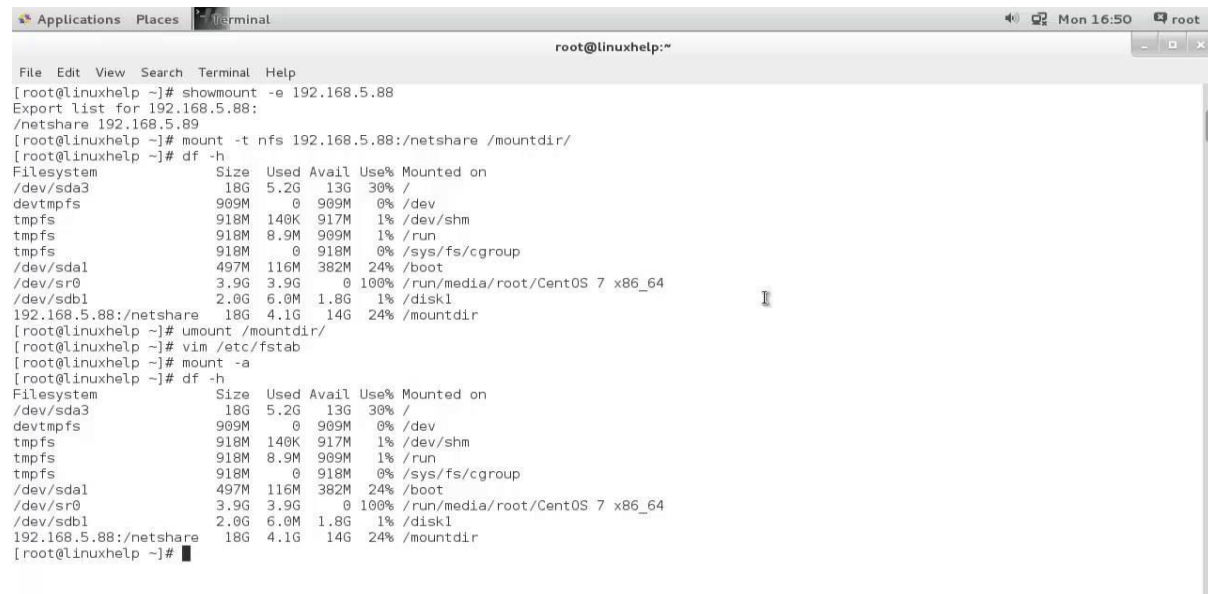
Options:-

-h = prints help message

-V = print a version string

Example:- mount /cd

OUTPUT



```
root@linuxhelp:~  
[root@linuxhelp ~]# showmount -e 192.168.5.88  
Export list for 192.168.5.88:  
/netshare 192.168.5.88  
[root@linuxhelp ~]# mount -t nfs 192.168.5.88:/netshare /mountdir/  
[root@linuxhelp ~]# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/sda3        18G  5.2G   13G   30% /  
devtmpfs         909M    0   909M    0% /dev  
tmpfs            918M  140K   917M    1% /dev/shm  
tmpfs            918M   8.9M   909M    1% /run  
tmpfs            918M    0   918M    0% /sys/fs/cgroup  
/dev/sda1        497M  116M   382M   24% /boot  
/dev/sr0         3.9G   3.9G    0 100% /run/media/root/CentOS 7 x86_64  
/dev/sdb1        2.0G   6.0M   1.8G    1% /disk1  
192.168.5.88:/netshare 18G  4.1G   14G   24% /mountdir  
[root@linuxhelp ~]# umount /mountdir/  
[root@linuxhelp ~]# vim /etc/fstab  
[root@linuxhelp ~]# mount -a  
[root@linuxhelp ~]# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/sda3        18G  5.2G   13G   30% /  
devtmpfs         909M    0   909M    0% /dev  
tmpfs            918M  140K   917M    1% /dev/shm  
tmpfs            918M   8.9M   909M    1% /run  
tmpfs            918M    0   918M    0% /sys/fs/cgroup  
/dev/sda1        497M  116M   382M   24% /boot  
/dev/sr0         3.9G   3.9G    0 100% /run/media/root/CentOS 7 x86_64  
/dev/sdb1        2.0G   6.0M   1.8G    1% /disk1  
192.168.5.88:/netshare 18G  4.1G   14G   24% /mountdir  
[root@linuxhelp ~]#
```

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 file.txt

OUTPUT

[illegible]

QUESTION 34:-WHAT IS USE OF “MORE” COMMAND?

Display text, one screen at a time

Syntax: - more [-dlfpsu][-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

[illegible]

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 file 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

```
ayush@negiPC28:/mnt/d$ cut -b 1-2 file.txt
fi
ayush@negiPC28:/mnt/d$ cut -c 1-3 file.txt
fil
ayush@negiPC28:/mnt/d$ |
```


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:

```
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE.BIN'      Downloads 2022  Pqdom  file.txt  newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  MISC      'System Volume Information'  file2.txt  testfile
ayush@negIPC28:/mnt/d$ gzip file.txt
ayush@negIPC28:/mnt/d$ ld
ld: no input files
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE.BIN'      Downloads 2022  Pqdom  file.txt.gz  newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  MISC      'System Volume Information'  file2.txt  testfile
ayush@negIPC28:/mnt/d$ gzip file.txt.gz
gzip: file.txt.gz already has .gz suffix -- unchanged
ayush@negIPC28:/mnt/d$ gunzip file.txt
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE.BIN'      Downloads 2022  Pqdom  file.txt  newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  MISC      'System Volume Information'  file2.txt  testfile
```

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:

```
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE_BIN'      'Downloads 2022'  'Pgdm'            file.txt           newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  'MISC'            'System Volume Information'  file2.txt          testfile
ayush@negIPC28:/mnt/d$ bzip2 file.txt
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE_BIN'      'Downloads 2022'  'Pgdm'            file.txt.bz2       newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  'MISC'            'System Volume Information'  file2.txt          testfile
ayush@negIPC28:/mnt/d$ bunzip2 file.txt.bz2
ayush@negIPC28:/mnt/d$ ls
'$RECYCLE_BIN'      'Downloads 2022'  'Pgdm'            file.txt           newfie.txt
'CLASSIFICATION OF COMPUTERS.pptx'  'MISC'            'System Volume Information'  file2.txt          testfile
```

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:

```
ayush@negiPC28:/mnt/d$ zip ab.tar file.txt file2.txt
  adding: file.txt (stored 0%)
  adding: file2.txt (stored 0%)
ayush@negiPC28:/mnt/d$ unzip ab.tar
Archive:  ab.tar
replace file.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  extracting: file.txt
replace file2.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  extracting: file2.txt
```

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:

```
ayush@negiPC28:/mnt/d$ tar -cvf a.tar file.txt file2.txt
file.txt
file2.txt
ayush@negiPC28:/mnt/d$ tar -xvf a.tar file.txt file2.txt
file.txt
tar: file.txt: time stamp 2023-06-11 20:12:19 is 26717.743307651 s in the future
file2.txt
ayush@negiPC28:/mnt/d$ tar -tvf a.tar file.txt file2.txt
-rwxrwxrwx ayush/ayush      10240 2023-06-11 20:12 file.txt
-rwxrwxrwx ayush/ayush        5 2023-06-07 21:31 file2.txt
ayush@negiPC28:/mnt/d$ |
```

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:

```
ayush@negiPC28:/mnt/d$ pr +1 file.txt

2023-06-11 20:21                                file.txt                                Page 1

Hello World
```

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 '-', Print all but the last K lines of each file.)

OUTPUT:

```
ayush@negiPC28:/mnt/d$ head -n 10 file.txt
Hello
World
This
is
file
where
i
am
writing
the
```

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:

```
ayush@negiPC28:/mnt/d$ tail -n 10 file.txt
World
This
is
file
where
i
am
writing
the
document
```

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:

```
ayush@negiPC28:/mnt/d$ paste file.txt file2.txt
Hello    file2
World
This
is
file
where
i
am
writing
the
document
```


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

```
ayush@negiPC28:/mnt/d$ sort file2.txt
23453454
6879504
69404664
ayush@negiPC28:/mnt/d$ sort -n file2.txt
6879504
23453454
69404664
ayush@negiPC28:/mnt/d$ sort -r file2.txt
69404664
6879504
23453454
```

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:

```
ayush@negiPC28:/mnt/d$ uniq file2.txt
23453454
321
6879504
69404664
```

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:

```
ayush@negiPC28:/mnt/d$ tr h H < file.txt
Hello
World
THis
is
file
wHere
i
am
writing
tHe
document
```

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:

```
ayush@negiPC28:/mnt/d$ cat file.txt
Hello
World
This
is
file
where
i
am
writing
the
document
ayush@negiPC28:/mnt/d$ cat file2.txt
23453454
23453454
321
321
6879504
69404664
ayush@negiPC28:/mnt/d$ cmp file.txt file2.txt
file.txt file2.txt differ: byte 1, line 1
ayush@negiPC28:/mnt/d$ comm file.txt file2.txt
      23453454
      23453454
      321
      321
      6879504
      69404664

Hello
World
comm: file 1 is not in sorted order
This
is
file
where
i
am
writing
the
document
comm: input is not in sorted order
```

```
ayush@negiPC28:/mnt/d$ diff file.txt file2.txt
1,11c1,6
< Hello
< World
< This
< is
< file
< where
< i
< am
< writing
< the
< document
---
> 23453454
> 23453454
> 321
> 321
> 6879504
> 69404664
\ No newline at end of file
```

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:

```
ayush@negiPC28:/mnt/d$ cat file.txt
Hello
World
This
is
file
where
i
am
writing
the
document
ayush@negiPC28:/mnt/d$ cat file2.txt
23453454
23453454
321
321
6879504
69404664
ayush@negiPC28:/mnt/d$ cmp file.txt file2.txt
file.txt file2.txt differ: byte 1, line 1
```

```
ayush@negiPC28:/mnt/d$ comm file.txt file2.txt
      23453454
      23453454
      321
      321
      6879504
      69404664
Hello
World
comm: file 1 is not in sorted order
This
is
file
where
i
am
writing
the
document
comm: input is not in sorted order
```

```
ayush@negiPC28:/mnt/d$ diff file.txt file2.txt
1,11c1,6
< Hello
< World
< This
< is
< file
< where
< i
< am
< writing
< the
< document
---
> 23453454
> 23453454
> 321
> 321
> 6879504
> 69404664
\ No newline at end of file
```

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:

```
ayush@negiPC28:/mnt/d$ cat file.txt
Hello
World
This
is
file
where
i
am
writing
the
document
ayush@negiPC28:/mnt/d$ cat file2.txt
23453454
23453454
321
321
6879504
69404664ayush@negiPC28:/mnt/d$ cmp file.txt file2.txt
file.txt file2.txt differ: byte 1, line 1
```

```
ayush@negiPC28:/mnt/d$ comm file.txt file2.txt
      23453454
      23453454
      321
      321
      6879504
      69404664
Hello
World
comm: file 1 is not in sorted order
This
is
file
where
i
am
writing
the
document
comm: input is not in sorted order
```



```
ayush@negiPC28:/mnt/d$ diff file.txt file2.txt
1,11c1,6
< Hello
< World
< This
< is
< file
< where
< i
< am
< writing
< the
< document
---
> 23453454
> 23453454
> 321
> 321
> 6879504
> 69404664
\ No newline at end of file
```

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

Ans. This command is used as interactive spell checker.

SYNTAX: aspell [check] [file_name]

OUTPUT:

aspell -c file.txt

```
Hello
World
Tis
is
file
where
i
am
writing
the
document

1) Ti's          6) Tips
2) Ties          7) Tits
3) Its           8) This
4) Tics          9) Ts
5) Tins          0) Tia
i) Ignore       I) Ignore all
r) Replace      R) Replace all
a) Add          l) Add Lower
b) Abort        x) Exit

? |
```

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:

```
ayush@negiPC28:/mnt/d$ cat file.txt  
This is a file  
this is a file  
THIS is a file  
THIS is not a file  
ayush@negiPC28:/mnt/d$ grep THIS file.txt  
THIS is a file  
THIS is not a file  
ayush@negiPC28:/mnt/d$ grep -i THIS file.txt  
This is a file  
this is a file  
THIS is a file  
THIS is not a file  
ayush@negiPC28:/mnt/d$ grep -w THIS file.txt  
THIS is a file  
THIS is not a file  
ayush@negiPC28:/mnt/d$ grep -e is -e a file.txt  
This is a file  
this is a file  
THIS is a file  
THIS is not a file
```

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:

```
ayush@negiPC28:/mnt/d$ sed 's/Unix/Linux/' file2.txt
Linux is great it's much better than windows
```

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

```
ayush@negiPC28:/mnt/d$ history
 1  ls -al
 2  help
 3  man pwd
 4  info -h
 5  info -a grep
 6  info -w grep
 7  whatis whatis
 8  mkdir
 9  type mkdir
10  type -a mkdir
11  w
12  w -h
13  w -o
14  w -u
15  date
16  date +%z
17  date +%x
18  echo "7+3" | bc
19  sudo apt install bc
20  echo "7+3" | bc
21  echo "var=20;var" | bc
22  cal -y 2023
23  sudo apt install ncal
24  cal -y 2023
25  sudo apt install ncal
26  who
27  who -Hu
28  q
```

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

```
ayush@negiPC28:/mnt/d$ cat file.txt
This is a file

ayush@negiPC28:/mnt/d$ cat file2.txt
Unix is great it's much better than windows
ayush@negiPC28:/mnt/d$ wc -l file2.txt | tee -a file.txt
1 file2.txt
ayush@negiPC28:/mnt/d$ cat file.txt
This is a file

1 file2.txt
```

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

```
ayush@negiPC28:/mnt/d$ cat file.txt
This is a file

ayush@negiPC28:/mnt/d$ cat file2.txt
Unix is great it's much better than windows
ayush@negiPC28:/mnt/d$ wc -l file2.txt | tee -a file.txt
1 file2.txt
ayush@negiPC28:/mnt/d$ cat file.txt
This is a file

1 file2.txt
ayush@negiPC28:/mnt/d$ umask
0022
ayush@negiPC28:/mnt/d$ umask 543
ayush@negiPC28:/mnt/d$ umask
0543
```

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

```
ayush@negiPC28:/mnt/d$ ls -l
ls: 'System Volume Information': Permission denied
total 764
drwxrwxrwx 1 ayush ayush      512 Jul 11  2022 '$RECYCLE.BIN'
-rwxrwxrwx 1 ayush ayush 755044 Aug  9  2022 'CLASSIFICATION OF COMPUTERS.pptx'
drwxrwxrwx 1 ayush ayush      512 Jun 12  2023 'Downloads 2022'
drwxrwxrwx 1 ayush ayush      512 Jun  9  22:49 MISC
drwxrwxrwx 1 ayush ayush      512 Jul 15  2022 Pydon
d--x--x--x 1 ayush ayush      512 Feb 24  23:24 'System Volume Information'
-rwxrwxrwx 1 ayush ayush 20480 Jun 11  2023 a.tar
-rwxrwxrwx 1 ayush ayush   326 Jun 11  18:41 ab.tar
-rwxrwxrwx 1 ayush ayush    30 Jun 13  2023 file.txt
-rwxrwxrwx 1 ayush ayush    45 Jun 13  2023 file2.txt
-rwxrwxrwx 1 ayush ayush    47 Jun 12  2023 file3.txt
-rwxrwxrwx 1 ayush ayush     5 Jun  8  11:11 newfie.txt
drwxrwxrwx 1 ayush ayush      512 Jun  8  11:42 testfile

ayush@negiPC28:/mnt/d$ chmod +t file2.txt
ayush@negiPC28:/mnt/d$ ls -l
ls: 'System Volume Information': Permission denied
total 764
drwxrwxrwx 1 ayush ayush      512 Jul 11  2022 '$RECYCLE.BIN'
-rwxrwxrwx 1 ayush ayush 755044 Aug  9  2022 'CLASSIFICATION OF COMPUTERS.pptx'
drwxrwxrwx 1 ayush ayush      512 Jun 12  2023 'Downloads 2022'
drwxrwxrwx 1 ayush ayush      512 Jun  9  22:49 MISC
drwxrwxrwx 1 ayush ayush      512 Jul 15  2022 Pydon
d--x--x--x 1 ayush ayush      512 Feb 24  23:24 'System Volume Information'
-rwxrwxrwx 1 ayush ayush 20480 Jun 11  2023 a.tar
-rwxrwxrwx 1 ayush ayush   326 Jun 11  18:41 ab.tar
-rwxrwxrwx 1 ayush ayush    30 Jun 13  2023 file.txt
-rwxrwxrwx 1 ayush ayush    45 Jun 13  2023 file2.txt
-rwxrwxrwx 1 ayush ayush    47 Jun 12  2023 file3.txt
-rwxrwxrwx 1 ayush ayush     5 Jun  8  11:11 newfie.txt
drwxrwxrwx 1 ayush ayush      512 Jun  8  11:42 testfile
```



```
ayush@negiPC28:/mnt/d$ chmod 4777 file2.txt
ayush@negiPC28:/mnt/d$ ls -l
ls: 'System Volume Information': Permission denied
total 764
drwxrwxrwx 1 ayush ayush      512 Jul 11  2022 '$RECYCLE.BIN'
-rwxrwxrwx 1 ayush ayush 755044 Aug  9  2022 'CLASSIFICATION OF COMPUTERS.pptx'
drwxrwxrwx 1 ayush ayush      512 Jun 12  2023 'Downloads 2022'
drwxrwxrwx 1 ayush ayush      512 Jun  9  22:49 'MISC'
drwxrwxrwx 1 ayush ayush      512 Jul 15  2022 'Pgdown'
d--x--x--x 1 ayush ayush      512 Feb 24  23:24 'System Volume Information'
-rwxrwxrwx 1 ayush ayush 20480 Jun 11  2023 'a.tar'
-rwxrwxrwx 1 ayush ayush   326 Jun 11  18:41 'ab.tar'
-rwxrwxrwx 1 ayush ayush    30 Jun 13  2023 'file.txt'
-rwxrwxrwx 1 ayush ayush    45 Jun 13  2023 'file2.txt'
-rwxrwxrwx 1 ayush ayush    47 Jun 12  2023 'file3.txt'
-rwxrwxrwx 1 ayush ayush     5 Jun  8  11:11 'newfie.txt'
drwxrwxrwx 1 ayush ayush      512 Jun  8  11:42 'testfile'
```

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:

```
ayush@negiPC28:/mnt/d$ ps
  PID TTY          TIME CMD
  398 pts/1        00:00:00 bash
  843 pts/1        00:00:00 ps
    :/mnt/d$ |
```

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

```
ayush@negiPC28:/mnt/d$ pstree
init--init--init--bash
      |--init--init--bash--pstree
      | 2*[{init}]
ayush@negiPC28:/mnt/d$ |
```

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 ...]

```
ayush@negiPC28:/mnt/d$ nice ps -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	1000	398	397	0	80	0	-	1520	do_wai	pts/1	00:00:00	bash
0	R	1000	846	398	0	90	10	-	1869	-	pts/1	00:00:00	ps

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.

```
top - 19:36:44 up 1 day, 7:20, 0 users, load average: 0.00, 0.00, 0.00
Tasks:  8 total,  1 running,  7 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 3829.3 total, 3612.7 free,  70.6 used, 145.9 buff/cache
MiB Swap: 1024.0 total, 1024.0 free,  0.0 used. 3572.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	1840	1180	1108	S	0.0	0.0	0:00.10	init
61	root	20	0	2172	364	0	S	0.0	0.0	0:00.00	init
62	root	20	0	2180	364	0	S	0.0	0.0	0:00.13	init
63	ayush	20	0	6080	5216	3492	S	0.0	0.1	0:00.24	bash
396	root	20	0	2188	380	0	S	0.0	0.0	0:00.00	init
397	root	20	0	2188	380	0	S	0.0	0.0	0:00.43	init
398	ayush	20	0	6080	5192	3468	S	0.0	0.1	0:00.73	bash
847	ayush	20	0	7788	3780	3188	R	0.0	0.1	0:00.03	top

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

```
ayush@negiPC28:/mnt/d$ 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"  
Enter the height and the base of the triangle :  
7 9  
Area of a triangle is 31.50
```

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

```
ayush@negiPC28:/mnt/d$ 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"  
Enter the length of the rectangle:  
20  
Enter the width of the rectangle:  
10  
The area of the rectangle is: 200
```

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

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

OUTPUT

```
ayush@negiPC28:/mnt/d$ echo "Enter side of square"
read num
area=$(expr "$num" \* "$num")
echo "The area of square is $area"
Enter side of square
25
The area of square is 625
```


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:

```
ayush@negiPC28:/mnt/d$ echo "Enter the radius of a circle :"  
read r  
area=$(echo "scale=2;3.14*($r*$r)" | bc)  
echo "Area of circle is $area"  
Enter the radius of a circle :  
7  
Area of circle is 153.86
```

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:

```
ayush@negiPC28:/mnt/d$ echo "Enter a number: "  
read num  
  
if (( num % 2 == 0 ))  
then  
    echo "$num is even."  
else  
    echo "$num is odd."  
fi  
Enter a number:  
4  
4 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 >=75 and <90...Excellent

If marks >=60 and <75...Very Good

If marks >=50 and <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

```
ayush@negIPC28:/mnt/d$ 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
Enter student's marks:
55
Good
```

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

```
ayush@negiPC28:/mnt/d$ 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  
Enter a year:  
2024  
2024 is a leap year  
:/mnt/d$ |
```

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

```
ayush@negiPC28:/mnt/d$ 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
Enter the first number: 23
Enter the second number: 43
Enter the third number: 32
43 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

```
ayush@negiPC28:/mnt/d$ 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
QUESTION: command not found
Enter the first number: 45
Enter the second number: 54
54 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

```
ayush@negiPC28:/mnt/d$ 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."
Enter the temperature in Fahrenheit: 78
78 degrees Fahrenheit is equal to 25.55 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 a number of month (1-12): " month
case $month in
  1)
    echo "January"
    ;;
  2)
    echo "February"
    ;;
  3)
    echo "March"
    ;;
  4)
    echo "April"
    ;;
  5)
    echo "May"
    ;;
  6)
    echo "June"
    ;;
  7)
    echo "July"
    ;;
  8)
    echo "August"
    ;;
  9)
    echo "September"
    ;;
  10)
    echo "October"
    ;;
  11)
    echo "November"
    ;;
  12)
    echo "December"
    ;;
  *)
    echo "Invalid input!"
    ;;
esac
```


OUTPUT

```
Enter a number of month (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

```
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: 3
Enter the second number: 4
The sum of 3 and 4 is 7.
-----
Please choose an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Quit
Enter your choice [1-5]: 2
Enter the first number: 4
Enter the second number: 3
The difference between 4 and 3 is 1.
-----
Please choose an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Quit
Enter your choice [1-5]: 3
```

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

```
ayush@negiPC28:~$ 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
Enter a character: d
d is a lowercase letter.
```

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

```
Current date and time:
Sun Jun 11 20:21:30 IST 2023
-----
List of files in the current directory:
$RECYCLE.BIN      Downloads 2022    'System Volume Information'  file.txt  newfie.txt
73.sh            MISC            a.tar              file2.txt  testfile
'CLASSIFICATION OF COMPUTERS.pptx'  Pgdom          ab.tar            file3.txt  tri.sh
-----
List of running processes:
  PID TTY          TIME CMD
 1036 pts/1        00:00:00 bash
 1054 pts/1        00:00:00 ps
-----
Execution time for 'ls' command:
$RECYCLE.BIN      Downloads 2022    'System Volume Information'  file.txt  newfie.txt
73.sh            MISC            a.tar              file2.txt  testfile
'CLASSIFICATION OF COMPUTERS.pptx'  Pgdom          ab.tar            file3.txt  tri.sh

real    0m0.011s
user    0m0.004s
sys     0m0.000s
```

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

```
ayush@negiPC28:/mnt/d$ read -p "Enter a positive integer: " n

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

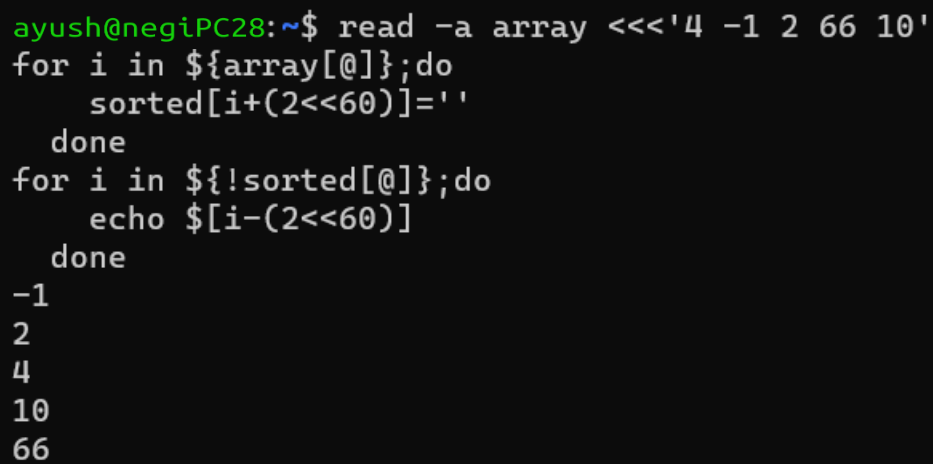
sum=0
for (( i=1; i<=$n; i++ ))
do
    sum=$(( sum + i ))
done

echo "The sum of the first $n natural numbers is $sum."
Enter a positive integer: 10
The sum of the first 10 natural numbers is 55.
```

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

```
read -a array <<<'4 -1 2 66 10'
for i in ${array[@]};do
    sorted[i+(2<<60)]=""
done
for i in ${!sorted[@]};do
    echo ${i-(2<<60)}
done
```

OUTPUT

A terminal window with a black background and green text. The prompt is 'ayush@negiPC28:~\$'. The user enters a shell script to sort the numbers 4, -1, 2, 66, and 10. The script uses an array, a loop to initialize sorted slots, and another loop to print the sorted values. The output shows the numbers sorted in ascending order: -1, 2, 4, 10, and 66.

```
ayush@negiPC28:~$ read -a array <<<'4 -1 2 66 10'
for i in ${array[@]};do
    sorted[i+(2<<60)]=""
done
for i in ${!sorted[@]};do
    echo ${i-(2<<60)}
done
-1
2
4
10
66
```

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

```
ayush@negiPC28:~$ 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
Enter a positive integer: 34
34 is not 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

```
ayush@negiPC28:~$ 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."
Enter a positive integer: 5
The factorial of 5 is 120.
```

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

```
ayush@negiPC28:~$ 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
Enter the number of terms in the Fibonacci series: 5
The Fibonacci series of 5 terms is:
0 1 1 2 3ayush@negiPC28:~$ |
```

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

```
Enter the number of terms in the Fibonacci series: 5
The Fibonacci series of 5 terms is:
0 1 1 2 3 ayush@read -p "Enter a positive integer: " ninteger: " 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
Enter a positive integer: 5
Multiplication table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

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

```
ayush@negiPC28:~$ # 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
Enter a string:
Hello World
Enter a word to find:
World
The word "World" was found at position 6 in the string "Hello World"
```

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

```
ayush@negiPC28:/mnt/d$ 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"
Enter the file name:
file2.txt
Word count: 8
Character count: 45
Space count: 7
```

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

```
ayush@negiPC28:/mnt/d$ touch result.txt
ayush@negiPC28:/mnt/d$ cat < calc.txt > result.txt
ayush@negiPC28:/mnt/d$ cat result.txt
2*3
2+3

ayush@negiPC28:/mnt/d$ |
```

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  
$ wcno file 2> errorfile
```

OUTPUT

```
ayush@negiPC28:~$ who  
ayush pts/3 2023-06-11 20:58  
ayush pts/3 2023-06-11 20:58  
ayush@negiPC28:~$ who > whofile.txt  
ayush@negiPC28:~$ cat whofile.txt  
ayush pts/3 2023-06-11 20:58  
ayush pts/3 2023-06-11 20:58  
ayush@negiPC28:~$ wc -l < whofile.txt  
2  
ayush@negiPC28:~$ touch nxtwhofile > whofile.txt  
ayush@negiPC28:~$ cat whofile.txt > nxtwhofile  
ayush@negiPC28:~$ 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

```
ayush@negiPC28:~$ wc -l whofile.txt
0 whofile.txt
ayush@negiPC28:~$ who | wc -l
2
ayush@negiPC28:~$ |
```

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

| = 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

```
ayush@negiPC28:/mnt/d$ vi file.txt
ayush@negiPC28:/mnt/d$ cat fie.txt
cat: fie.txt: No such file or directory
ayush@negiPC28:/mnt/d$ cat file.txt
This is a file
Editing in VI Editor
1 file2.txt
```



(VI editor in action)

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

```
ayush@negiPC28:~$ su
Password:
su: Authentication failure
ayush@negiPC28:~$ sudo su
[sudo] password for ayush :
root@negiPC28: /home/ayush# |
```

QUESTION 89. What is the “awk” ?

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.

Default Behaviour

Syntax : awk '{print}' <filename>

```
ayush@negiPC28:~$ su
Password:
su: Authentication failure
ayush@negiPC28:~$ sudo su
[sudo] password for ayush :
root@negiPC28: /home/ayush# |
```

Print the lines which match the given pattern

Syntax : awk '<pattern> {print}' <filename>

```
ayush@negiPC28:/mnt/d$ awk '/is/{print}' file2.txt
Unix is great it's much better than windows
:/mnt/d$ |
```

Splitting a Line Into Fields

For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the \$n variables. If the line has 4 words, it will be stored in \$1, \$2, \$3 and \$4 respectively. Also, \$0 represents the whole line.

Syntax : awk '{print \$<1> , <2> , <n> }' <filename>

```
ayush@negiPC28:/mnt/d$ awk '{print $1, $4}' file2.txt
Unix it's
```

QUESTION 90. What is the “at” command ?

In Linux, the at command is used to schedule one-time tasks to be executed at a specified time in the future. It allows you to submit commands or scripts that will run in the background without any further interaction from you.

Syntax: at [-f file] [-l] [-q queue] [-v] time

Once you enter the at command, you will be presented with a prompt where you can enter the commands or scripts that you want to schedule for execution. Press **Ctrl+D** to indicate the end of input and submit the job.

(at seems to be disabled , by the OS due to security reasons)