=================== What is Shell Scripting ===================
A shell script is a text file that contains a sequence of commands
Or called as command-line interpreter.
The shell is the operating system's command interpreter and the set of commands you use to communicate with the system.

Everything is a file, Linux and other Unix-like Operating systems maintain a consistency by treating everything
as a file (even the hardware devices). The keyboard, mouse, printers, monitor, hard disk, processes,
even the directories are treated as files in Linux.

=================== Who Required Shell Scripting ===================
Linux Administrators => Automate thier work for writing the networking commands
Programmers => Use to make a installer of a programs
Application Developers
Application Tester => Automate thier testing work

=================== Examples of shell script applications ===================
Automating the code compiling process
Running a program or creating a program environment
Completing batch (what is batch file)(A batch file is a script file that stores commands to be executed in a sequence or in order)
Manipulating files. (Changing the file permissions)
Linking existing programs together. (Merging/concatenate 2 different files)
Executing routine backups. (take a backup of specific directories)
Monitoring a system (Network/harware level monitoring)

=================== Advantages and disadvantages of shell scripts ===================
======== Advantage ========
Shell scripting is meant to be simple and efficient.
It uses the same syntax in the script as it would on the shell command line, removing any interpretation issues. (Like Compilers, Asemblers, Likers)
Writing code for a shell script is also faster and requires less of learning curve than other programming languages.
======== Disadvantage ========
However, if there is an error in a shell script, this can prove to be extremely costly if left unnoticed.
Additionally, differing platforms associated with shell scripting may not be compatible.
Shell scripts can also be slower to execute than individual commands.

=================== LINUX Directory Structure ===================
Open the HOME dir and list the contents
to goto home directory we uses "/"
This is the place where your LINUX OS is been installed

/bin directory:- ALL THE COMMAND WHICH WE EXECTUE IN THE TERMINAL ARE KEPT INSIDE THE /bin dir

eg of commands like cd, touch , mkdir, ls, ll, man, cp, mv... and etc...

/boot directory:- Conatins all the boot information like(from where to boot the os, and load the required drivers)

and the boot configuration file are also stored under boot directory

/dev directory:- where it store a special file such as DVD/CD, thumb drive like pendrive and etc...

/etc directory:- It contains all the configuration file for SSH, username & hostname

/home directory:- inside home one another home directory will be created so that the different user can store thier file/data/configurationfile and etc.....

whenever we create new user his name directory will be created under the home dir

/lib directories:- use for OS level usage to keep the lib file and so on

/media & /mnt is an optional directories where we can mount different directories like Pendrive, etc

/opt directories:- is an optional directory where we can install any third party softwares like google chrome. firefox mozilla and open office and etc...

/proc directories:- system memory information is stored here & CPU information will also be there

/root directories:- is to for root user to store everything

/sbin directories:- contains all the executable file for super user (like mount, su)

/tmp directories:- it will store the temporary files,

for eg:- if you have downloaded any software for testing purpose you can keep it here

once the system is restarted the temp folder will become empty

/usr directories:- very important directory where all the softwares are installed

/var directories:- (log information) contains all the information about the log of the system like startup time, error log, warning log and etc...

Example of Shellscripting file

For Ex:- create a basic text file with .sh as a extension name test.sh

```
vi test.sh
        #!/bin/bash

        mkdir createdbyscript
        touch createdbyscript/basic.txt

        echo "Testing Script File" >> createdbyscript/basic.txt
```

then change the permission to execute the program

```
chmod u+x test.sh
```

Run the program
sh test.sh

========================== Commands on SHELL ======================================

```
whoami     => Display the current logged in user
w                        => Display more detail of the current logged in users like Session id, system up time
                           how many user are logged in now, CPU utilization display
                                    At the end it also display who many user is logged in and from where like Graphic or putty
who                      => Display current logged in users
who -b      => system boot up time
uname       => OS information
uname -a => Display OS, Login username, Kernel version, current hardware support like x86 or x64
arch        => Display OS architecture like x86 or x64
hostname => Show/Set system hostname
hostname -f => Display fully qualified Domain name
hostname -i => Display ip address of the host


echo $PATH
cd ~ go directly to the home directory.
cd - will point you back to the place from you came from
cd          There are always 2 path while using 'cd' Relative & Absolute path
            1) Relative path => just give a name like this cd /SoftDev then cd DAC2022
            2) Absolute path => give the entire directory location like /SoftDev/DAC2022

date display the current date
            if we want to format the date then we can go for the following commands
                        date +%d-%m-%Y
                        date +%d:%:m%:Y

cal to display the calender of current month
            cal -y 2017 to display the months of 2017

touch       use to create empty files.
                        touch File{1..10}.txt
                        touch Test{1..10}.txt
                        touch alpha{a..z}.txt
                                    create file in sequences

head - output the first part of files
    Print  the  first  10  lines  of each FILE to
    standard output.

head -n 4 filename displays only 4 line from starting
```

tail - output the last part of files
Print the last 10 lines of each FILE to standard output.

tail -4 filename displays only last 4 line from the ending.

more - Give you the option to stay inside the terminal itself and read the file with large contents

cat filename | more - use | operator if you want to pass the output of one command to another

lpr filename ==> Press Enter and type the Message
Submits files for printing. Files named on the command line are sent to the named printer (or the default destination if no destination is specified). If no files are listed on the command-line, lpr reads the print file from the standard input.

cp /bin/???? . Copy the file/folder from /bin dir with 4 char
ls –a (create .anyfile for example)--all .a file show with name (do not ignore entries starting with .) << Create file with .name>>
DISPLAY ALL THE HIDDEN FILE IN THE CURRENT DIRECTORY
ls –p (--indicator-style=slash append / indicator to directories)
ls -r, --reverse (reverse order while sorting)
ls -R, --recursive (list subdirectories recursively) <<Create SUB DIR first>>
ls -l (List all the files) equivalent to "ll" command
ls -i (-i, --inode print the index number of each file) to view it type ==> stat --format=%i third
An inode is an entry in inode table, containing information (the metadata) about a regular file and directory.
Like (File type, Permissions to that file, Link count,User ID,Group ID,Size of file,Time stamp,Attributes,
Access control list & Other Meta-Data)
ls -o (like -l, but do not list group information) <<dont display USER column>>
ls --sort=time (List according to the time)
ls --sort=size (list according to the size)

touch a b c Easiest way to create an empty file
touch abc xyz lmn pqr nba gcr
mkdir -p MET MET/IIT MET/IIT/DAC MET/IIT/DACA MET/ISDR/ADSD (Create all subfolder in 1 shot)
ls –R (Recursive list subdirectories recursively)
MANUAL man ls, rm,

ls a (List all the file with name starting with ‘a’
ls a* p* l*
ls [apl]*
ls [!apl]* (List all the file which is not starting from a, p, l)

ls stt* (Just a couple of letter it will show the result)
ls > output (All the cotent of ls will be stored in the the file with name output)

============================ apropos ==============================
The apropos command displays a list of all topics in the MAN pages (i.e., the standard manual that is built into Unix-like operating systems) that are related to the subject of a query.
apropos takes its name from the English word with the same spelling (and the same pronunciation) that means relevant. It is particularly useful when searching for commands without knowing their exact names.
apropos's syntax is:
apropos keywords
apropos search
man find grep

Cat(concatenate) command is very frequently used in linux.
It reads data from file and give their content as output to the terminal(STD output) or file.
It helps us to create,view,concatenate files.
So let us see some frequently used cat commands.

Cat filename (cat - concatenate files and print on the standard output)
cat [OPTION] [FILE]...
```
cat > first                 => Create a File with Cat Command
cat first                   => just display the file contents in terminal
cat >> first         => Appending The file Double Redirection Operator
cat < first          => just display the file contents ((OR)) it use file name first as a
input for a command and output will be shown in a terminal.
cat first second     => View Contents of Multiple Files in terminal
          OR
cat test; cat test1; cat test2      => View Contents of Multiple Files in terminal
cat first second third > fourth => Redirecting Multiple Files Contain in a Single File
```

Use below to sort into alphabetical & put output to another file
CAT TEST TEST1 TEST2 TEST3 | SORT > TEST4 => SORTING CONTENTS OF MULTIPLE FILES IN A SINGLE FILE

Cat -E, (--show-ends display $ at end of each line)
Cat -T, (--show-tabs display TAB characters as ^I)
Cat -n, (--number number all output lines
Cat -s, (-- squeeze-blank) Cat command can suppress repeated EMPTY LINES in output like $cat -s geeks.txt

Re-direction
```
ls > output
whoami > username    => put the content into username file
cat username                => Display the file
date > datefile             => put the content into username file
cat datefile         => Display the file
history > historyfile
cal > calenderF
-----
```

man ls > ManualS
col -b < ManualS   => Do Not output any backspaces, printing only the last character written to each column position
col -b < ManualLS > Man-LS
cat ManualS | more   => Use Cat Command with More & Less Options to get fitted in terminal. Go Till the end to quit
cat ManualS | less   => Less Options IN "less" we can ues Q to quit anytime

============== tac ========================
tac - concatenate and print files in reverse order Last will come first

============= Standard File Descriptors =============
The file descriptors 0, 1, 2 are kept for the bash shell usage.
0 => STDIN   => STDIN stands for standard input which is the keyboard by default.
1 => STDOUT => This stands for the standard output which is the screen by default.You can redirect output to a file using the >> symbol.
2 => STDERR => This file descriptor is the standard error output of the shell which is sent to the screen by default.
If you need to redirect the errors to a log file instead of sending it to the screen, you can redirect errors using the redirection symbol
You can use the above file descriptors to control input and output.

//pwd >> myfile
cat first second third   => Cat Multiple File
cat first second third 2> errlogfile=> Error Will be stored in errlogfile
cat first second third 2>> errlogfile   => Error Will be APPENDED & stored in errlogfile
ls -l first second tenth 2>> errorlogfile 1> myfile ==> Execute the ls -l command if error is there it will be redirected to errorlogfile & result will be redirect to myfile
ERROR
FILE   RESULT FILE

============================= • PATTERN MATCHING =============================
GREP => Use for pattern matching.
The grep filter searches a file for a particular pattern of characters,
and displays all lines that contain that pattern.
The pattern that is searched in the file is referred to as the regular expression
(grep stands for globally search for regular expression and print out)

man grep
=> Display Manual of Grep usage in UNIX
man grep > grepfile   => Redirect the content of grep into grepfile
vi grepfile
col -b < grepfile   => Do not output any backspaces, printing only the last character

written to each column position.
col -b < grepfile > grepopfile => Properly output the file and put it into grepopfile
vi grepopfile
grep -e beginning grepopfile => -e For expression Searches for "beginning" in file
grep -i symbolic grepopfile => Ignoring case sensitivity
grep -c symbol grepopfile => Count the lines where strings are matched with -c option
grep -v symbol grepopfile => The -v option instructs grep to print all lines that do not contain or match the expression.
grep -vc symbol grepopfile => Count the lines that do not contain or match the expression.
grep -n symbol grepopfile => Return the Actual Lines Number that contain the search pattern with -n option
grep -w symbol grepopfile => Search for exact matching word using the -w option
grep -w "the named" grepopfile => Search for exact matching word using the -w option make Symboolic and do the comparision between -i and -w

^ Matches characters at the beginning of a line
$ Matches characters at the end of a line
"." Matches any character given in the range
[a-z] Matches any characters between A and Z
[^ ..] Matches anything apart from what is contained in the brackets

grep ^GREP grepopfile => To print lines beginning with a certain character
grep POSIX$ grepopfile first make POSIX in file at end of line => To display lines that end with the letter POSIX
grep pattern$ grepopfile => To display line Ending with "pattern" use $
grep -c pattern$ grepopfile => To return count of line returning "pattern"
grep "POSIX.)"$ grepopfile => If special character the use pair of ""
grep "P" grepopfile => "P" Matches any character in the line for "p" like pipe or program or pant and etc....
grep -c "p" grepopfile => Return count of line matchin the character "p"
grep -c [a-z] grepopfile => [a-z] Matches any characters between A and Z & return the line
grep [0-9]
=> [0-9] Matches any characters between 0 and 9 & return the line

grep -c [0-9]
=> [0-9] Matches any characters between 0 and 9 & return the count
grep -R students MET   => Search all files in the current directory and in all of its subdirectories for the word 'students' create name file with student keyword
grep --color binary grepopfile   => Finally, you can force grep to display output in colors
grep -w binary grepopfile   => Force PATTERN to match only whole words
grep -l binary grepopfile   => Print only names of FILEs with selected lines

--- WC ---
--

wc stands for word count. As the name implies, it is mainly used for counting purpose.
It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.
By default it displays four-columnar output.
First column shows number of lines present in a file specified
Second column shows number of words present in the file
Third column shows number of characters present in file
Fourth column itself is the file name which are given as argument.

wc grepopfile
=> wc stands for word count
wc state.txt capital.txt
Note : When more than file name is specified in argument then command will display four-columnar output for all individual files
plus one extra row displaying total number of lines, words and characters of all the files specified in argument,
followed by keyword total.

wc -l grepopfile   => displays two-columnar output, 1st column shows number of LINES present in a file and

2nd itself represent the file name.
wc -w grepopfile   => This option prints the number of WORDS present in a file.
wc -c grepopfile eg:-size <ls -l>   => This option displays count of BYTES present in a file.
wc -m grepopfile   => This option displays count of CHARACTERS from a file.
wc -L grepopfile   => used to print out the length of longest (number of characters) line in a file

grep -n -e beginning grepopfile => Searches the Expression & Display the line number where the "beginning" is written
grep -nc -e beginning grepopfile => Searches the Expression & COUNT the line consisting the expression "beginning" and return count
grep . grepopfile => Search for line with ATLEAT SINGLE OR MORE CHARACTERS
grep -c . grepopfile => Return the count of line having ATLEAT SINGLE OR MORE CHARACTERS
grep "\.$" grepopfile => Return line ends with (.) META DATA THATS Y USING "\"
grep -c "\.$" grepopfile => Return count of line ending with (.)
grep -c "\." grepopfile => Return count of . no of time it appears in the file

------------------------ CUT ------------------------------
cut - remove sections from each line of files

ls -l $filename | cut -c14-19
// -c, --characters=LIST => select only these characters
vi FileDetail > Group name of the file is
ls -l variabletest.sh | cut -c14-19 | cat >> FileDetail

-- TR ---

TR is use to translate or delete characters
tr "abc" "xyz" directly on the Shell
It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters
and basic find and replace.
It can be used with UNIX pipes to support more complex translation

ll > pract
vi pract
cat pract
tr " " "|" < pract or * => Replace space to | symbol
tr -s " " "|" < pract => squeeze repeated
tr -s =>use to squeeze repetition/occurence of charcter || remove repeated charcter
tr -s " " "|" < pract > spract =>squeeze repeated & put the output in spact file
cat spract
tr -d "0-9" < spract =>Delete the character given in range <<Delete between 0to9>>
tr -d "0-9 a-z" < spract =>Delete the character & Alphabets and display
tr -dc "a-z" < spract =>Dont Delete the given character and

display contineously

```
tr -dc "a-z\n" < spract                    =>Dont Delete between a to z and New Lines
tr -dc "a-z\012" < spract
tr "a-z" "A-Z" < spract                    =>           replace lower to capital letters
vi names
sort names                                                   =>Sort according to
alphabets
uniq names                                                              =>Remove
Immideate Duplication Name
sort names > snames

vi grepopfile
tr "a-z" "A-Z" < grepopfile
tr "a-z" "A-Z" < grepopfile > g1
vi g1
tr -dc "A-Z \012" < g1
tr -dc "A-Z \012" < g1 > g2                =>Dont Delete the given character and black
line & display contineously
tr " " "\012" < g2
tr " " "\012" < g2 > g3                              => Space is replace by new line
cat g3
grep . g3                                                    => Display Line which
is having ATLEAST 1 CHARACTER
grep . g3 > g4
grep ^...$ g4                                                => Only put 3
character in a line
grep ^...$ g4 > g5                                   => 3 Char per line
sort g5
sort g5 > g6
uniq g6
uniq g6 > finalopfile
================================== ALL ABOVE COMMANDS In SINGLE
LINE  ======================
cat grepopfile
cat grepopfile | tr "a-z" "A-Z"
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012"
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012"
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep .
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$ | sort
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$ | sort |
uniq
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$ | sort |
uniq > finalopfile
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$ | sort |
uniq tee  finalopfile
cat grepopfile | tr "a-z" "A-Z" | tr -dc "A-Z \012" | tr " " "\012" | grep . | grep ^...$ | sort |
uniq | tee  finalopfile
================================== In SINGLE LINE ALL ABOVE THINGS
SAME ======================
tee                                                          => Send the output to
```

the file and do the STD output on the screen

```
expr $x + $y            =>ADD
expr $x - $y            =>SUB
expr $x / $y            =>DIV
expr $x * $y            =>MUL NOT WORK
expr $x \* $y           =>MUL will work like this
```

==== SYSTEM VARIABLES ====

```
#! /bin/bash

echo Our Shell name is $BASH
echo Our Shell version is $BASH_VERSION
echo Our Home directory is $HOME
echo Our Current working directory is $PWD
```

```
vi hello.sh

vi template
mkdir script
vi script/hello.sh
vi script/quote.sh
vi script/countarg.sh
sh script/coutarg.sh *  => count the content of current dir with * wild card

cp script/coutarg.sh script/countarg1.sh
vi script/countarg2.sh
vi variabletest.sh
cp variabletest.sh iotest.sh
vi iotest.sh
cp iotest.sh iftest1.sh
vi iftest1.sh
cp iftest1.sh iftest2.sh
vi iftest2.sh
vi whiletest.sh
cp whiletest.sh fortest.sh
vi fortest.sh
vi foreachtest.sh
vi casetest.sh
vi filepermission.sh
```