Shell scripting

What is a shell?

* A shell is a program that takes commands from the user and gives them to the operating system to execute.
* It acts as a bridge b/w the user and the kernel

Common shells in Linux/unix:-
* sh → Bourne shell
* bash → Bourne again shell
* zsh, ksh, fish → other modern shells.

What is Bash?

* Bash = Bourne again shell
* It is the most widely used shell in linux.

Kernel:-

* The kernel is the core part of an operating system (OS).
* It acts a bridge b/w hardware & software.
* without kernel, your applications could not interact with your computer hardware.

To execute a file → ./sample-shell-script.sh

Commands:-

ls:-

lists files and folders in the current direct-ory.

ls -ltr:-

It shows all files in long format, sorted by time, but in reverse order → oldest first, newest last

How to create a file?

By using touch command

Man:-

The man command is short for manual. It shows the manual (documentation) for other commands.

We can use any command with man ls -

Touch:-

* It is used to create an empty file.
* for automation it is used.

vi (Visual Editor):-

* One of the oldest text editors in unix.
* Comes pre-installed in almost every unix/linux system.
* Lightweight, minimal features.
* works in diff modes:-
  (i) Command mode → for giving comm-ands.

(ii) Insert mode → for actual typing/editing level.

(iii) Last line mode → for commands like :wq (save & quit)

Vim (Vi improved) :-

* stands for Vi improved (an enhanced version of vi)
* Provides more features than vi, like:
  * Syntax highlighting
  * Undo/redo multiple times
  * Plugins and extensions support
  * Search & replace with patterns.
  * Better navigation and editing commands.
* It creates and opens the file

Shebang :-

(#!)

* A shebang is the character sequence (#!) at the very top of a script file.
* It tells the system which interpreter should be used to run the script.
* Without it, the script may not run properly.

Cat :-

* Cat stands for concatenate
* It is a command-line utility in Unix/Linux used to:-
  (i) create files
  (ii) view content of files
  (iii) Concatenate (join) multiple files
  (iv) Redirect output to another file.

Chmod :-

* Chmod stands for change mode.
* It is used to change the permissions of files and directories in Linux/unix.
* Permissions control who can read, write, or execute file.

File Permission in Linux :-

Every file/directory has 3 types of permissions:-

1. r → read
2. w → write
3. x → execute

and they apply to 3 categories of users:-

* u → user (owner)
* g → group
* o → others
* (a → all users = u+g+o)

Ex:- * owner: rwx → read, write, execute
* group: r-x → read, execute
* others: r-- → read only.

Using Chmod :-

There are two ways: symbolic mode and numeric mode

(1) Symbolic Mode:-

* Add or remove permissions

Examples:-

```
Chmod u+x file.sh # give execute permission to user.
Chmod g-w file.txt # remove write permission from group.
Chmod o+r file.txt # give read permission to others.
Chmod a+x script.sh # give execute permission to everyone.
```

(2) Numeric (octal) Mode:-

Permissions are represented as numbers:-

* r=4
* w=2
* x=1

Add them to set permissions:-

* 7 = rwx
* 6 = rw
* 5 = r-x
* 4 = r--

History:-

* The history command in Linux shows a list of previously executed commands in the current shell session.
* It's very useful for recalling, re-running, or analyzing past commands.

Pwd:-

* Pwd stands for Print working Directory.
* It shows the full path of the directory you are currently in.

options:-

(1) Pwd -L (logical path, default)
   * Prints the path as stored in your shell, even if it contains symbolic links.

(2) Pwd -P (Physical path)
   * Prints the actual physical path, resolving all symbolic links.

mkdir :-

* mkdir stands for make directory.
* It is used to create new directories (folders) in Linux / unix systems.

Useful options :-

(1) Create multiple directories :-

```
mkdir dir1 dir2 dir3
```

*Creates 3 folders at once.

(2) Create parent directories (-p) :-

```
mkdir -p projects/python/app
```

*Creates the entire path (projects, python, and app) if they don't already exist. without -p, it would throw an error if the parent directory is missing

(3) Set permissions while creating (-m) :-

```
mkdir -m 755 mydir
```

*Creates mydir with specific permissions (rwxr-xr-x)

Cd :-

* Cd stands for change directory
* It is used to move btw directories in the filesystem
* By default, when you open a terminal, you start in your home directory

Useful variations of cd:-

(1) Go to home directory:-

```
cd
 or
cd ~
```

(2) Go up one level:-

```
cd ..
```
* Moves from current directory to parent directory

(3) Go up multiple levels:-

```
cd ../../
```
* Moves up 2 levels

(4) Go to previous directory:-

```
cd -
```
* Switches back to the last directory you were in

(5) Absolute path:-

```
cd/var/log
```
* Moves to /var/log from anywhere

(6) Relative path:-

```
cd/projects/python
```
* Moves to projects/python inside your current directory.

rm:-

* rm stands for remove
* It is used to delete files and directories in Linux/unix
* Once deleted with rm, files cannot be recovered easily.

Useful options :-

(1) Delete multiple files :-

rm file1.txt file2.txt file3.txt

(2) Force delete (-f)

rm -f file.txt

* Deletes without asking for confirmation

(3) Delete directory with contents (-r) :-

rm -r myfolder

* Removes a directory and everything inside it.

(4) Force + recursive (-rf) :-

rm -rf myfolder

* Deletes a directory and all its content -s without any confirmation

(5) Interactive mode (-i) :-

rm -i file.txt.

* Asks for confirmation before deleting

Df :-

* df stands for disk filesystem
* It shows how much disk space is used and available on mounted filesystems.

Free :-

* free is used to display memory usage in Linux.
* It shows detail about * Ram (used, free, available)
* swap memory (used/free)

Nproc :-

* nproc stands for no. of processing units.
* It shows how many CPU cores (processing units) are available to the current process.

Top :-

* top shows a real-time view of running processes in Linux.
* It displays CPU usage, memory usage, running processes, load average, uptime, etc.
* It is like the Linux version of Task manag -er in Windows.

Echo :-

* echo is used to display text or variables on the terminal.
* It is mainly used in shell scripting to print messages, debug, or show values.

set -x :-

* In bash scripting, set -x enables a debuggi -ng mode.
* It makes the shell print each command before executing it, along with its arguments.
* This helps you see exactly what your script is doing.

ps :-

* ps stands for process status
* It shows a snapshot of running processes at the moment when you run the command.

Common options :-

1. Show all processes (-e or -A) :-

   ps -e

2. Show all processes with full details (-ef) :-

   ps -ef

3. Show processes for a specific user :-

   ps -u vivek

4. Tree view of processes (--forest) :-

   ps -ef --forest

ps -ef | grep :-

1. ps -ef :-

   * Shows a list of all processes in full detail (system-wide).
   * Columns include UID, PID, PPID, CMD, etc.

2. | (pipe) :-

   * Sends the output of the first command (ps -ef) as input to the second (grep).

③ grep :-

* searches (filters) the output for a given keyword.

→ Together, it is used to search for a specific running process.

Awk :-

* Awk is a text-processing and pattern scanning tool.
* It reads input line by line, splits it into fields (based on spaces or another delimiter), and lets you process or print specific fields.
* It is often used to extract columns from text, generate reports, and automate log analysis.

Important concepts :-

* $0 → entire line
* $1, $2, $3 ... → first, second, third field (column)
* NF → no. of fields in the line.
* NR → current line number
* FS → field separator (default : space or tab)

set -e :-

In bash/shell scripts,

* set -e tells the shell to exit immediately if any command fails (returns a non-zero status code).
* Without set -e, even if one command fails, the script continues executing the next commands — which can cause unexpected behaviour.

So, set -e makes your script safer by stopping at the first error.

set -o :-

* In bash, the set command controls the shell's behaviour.
* Using set -o <option> (or set +o <option> to disable), you can enable/disable diff. shell options.

Curl :-

* Curl = client URL
* It's a command-line tool to transfer data to/from a server using supported protocols like HTTP, HTTPS, FTP, SCP, SFTP, LDAP, SMTP, POP3, IMAP, etc.
* Widely used for API testing, downloading files, uploading data, and debugging network requests.

Wget :-

* wget = World wide web get.
* It is a non-interactive command-line tool used to download files from the web (HTTP, HTTPS, FTP).
* Unlike curl, which is mainly for data transfer and API testing, wget is specialized for downloading files, entire websites, and recursive downloads.

* Use wget → when you need to download
* Use curl → when you need to interact/test APIs or debug.

Sudo :-

* Sudo = Superuser do.
* It allows a permitted user to run commands as the root (Superuser) or another user.
* Commonly used to perform administrative tasks like installing software, modifying system files, restarting services, etc.

Su :-

* Su = Substitute user / switch user.
* It allows you to switch to another user account

find :-

* The find command is used to search for files and directories in a directory hierarchy.
* It can search based on name, type, size, permissions, modification time, owner, etc.
* You can also execute actions (like delete, move, run) on the results.

trap :-

* trap is a bash built-in command used to catch signals and errors in a shell script, and then run custom commands before the script exits.
* Signals are events like CTRL+C (SIGINT), kill (SIGTERM), or script exit (EXIT).
* With trap, you can clean up resources (like deleting temp files, stopping services) before a script ends.

Loops :-

* A loop is a structure that repeats a set of commands multiple times until a condition is met.

1. For loop
2. while loop
3. until loop
4. ~~If-else~~
5. elseif

① for loop :-

Example :- Loop over numbers

```
for i in 1 2 3 4 5
do
    echo "Number: $i"
done.
```

→ Print number 1 to 5.

② while loop :-

Example :- Count until 5

```
count =1
while [ $count -le 5]
do
    echo "count : $count"
    count =$(( count +1 ))
done
```

③ until loop :-

It runs until the condition becomes true.

```
count =1
until [ $ count -gt 5]
do
    echo "count : $count"
    count = $(( count +1 ))
done
```

(4) If-else loop :-

* if-else is a decision-making statement.
* It runs commands conditionally, depending on whether a condition is true or false.

Structure :-

```
if [ condition ]
then
    # commands if condition is true
else
    # commands if condition is false
fi
```

(5) if elsen (elif) loop :-

* Used when there are multiple conditions.

Ex :-

```
num=10
if [ $num -gt 15 ]
then
    echo "greater than 15"
elif [ $num -gt 5 ]
then
    echo "greater than 5 but less or equal 15"
else
    echo "5 or less"
fi
```