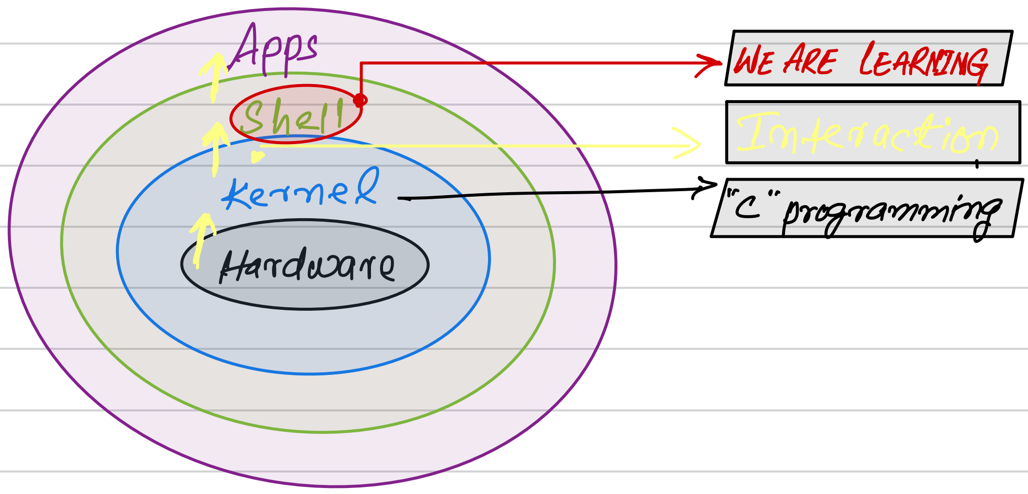
Day # 02 : (Shell Scripting)

INTRODUCTION TO SHELL AND ENVIRONMENT SETUP

Linux Os

Linux Architecture →



→ We operate on linux using shell commands.
↳ Example : We are printing something, shell command is `Echo`
↳ There are a lot of commands to communicate with linux operating system.

→ `/bin/sh` → path.
↓ Improvement in Shell (scripts).
↓
`/bin/bash` → mostly used !

→ To run shell Script → First we need an environment → can be provided by multiple cloud platforms.

→ Create an EC2 instance → where we will run shell scripts.

→ Connect with it
→ Install Text Editor to write a Shell Script.

(vim, vi nano etc)

Insert Mode (Type 'i')

To Exit
(:wq , :qa)
:wq → write & quit
:qa → quit without saving.

hello.txt
```
Hello Abdur
~
~
~
~
~
```

→ To create a new file `vim hello.txt` → (write something & Exit)
→ To veiw what's in the file.
`cat hello.txt` → "Hello Abdur".

Shell Script

→ Shell script extension "sh"
→ create a shell script name "hello" → `vim hello.sh`

```
#!/bin/bash
# This is Sample Script
echo "Hello Abdur"
echo " Hello DevOps"
echo "Abdur: create
      a directory"
echo "DevOps: Okay!"
mkdir sample.
```

#!/bin/bash → Tells interpreter which type of Shell it is —

→ Now if you do `ls -l` → To check permission.
↓ It will be read and write only (Not Exe

→ Now to make it executable,
run → `chmod 754 hello.sh`
↓ hello.sh is now executable.
↓ Run → `./hello.sh`

Basic Scripting Skills.

↳ variables , Arguments , Conditionals , Loops
Functions.

$$\begin{tikzcd} & \text{Loops} \arrow[dl, dash] \arrow[dr, dash] & \\ \text{for} & & \text{while} \end{tikzcd}$$

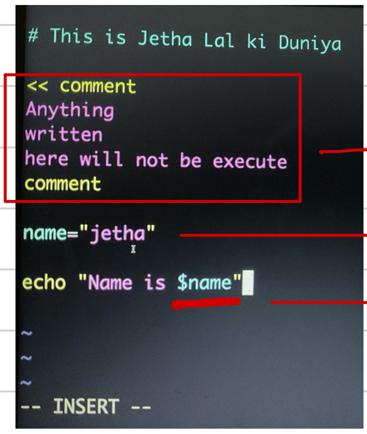
—VARIABLES

# → single line comment

<< (Block)
Anything
written
here → are commented!
(Block)

Example →

<< comment
Anything
written
here → are commented!
comment.



```
# This is Jetha Lal ki Duniya

<< comment
Anything
written
here will not be execute
comment

name="jetha"

echo "Name is $name"
~
~
~
-- INSERT --
```

→ Single Line Comment

→ Multi-line Comment

→ Variable

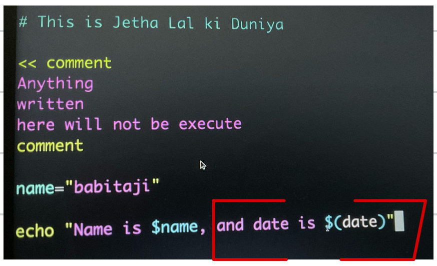
→ $ → dollar sign use for identification of variable.

↳ Script file → chmod 755 <file-name.sh>

↳ To run Script → ./<file-name.sh>

↓

Name is jetha → output



```
# This is Jetha Lal ki Duniya

<< comment
Anything
written
here will not be execute
comment

name="babitaji"

echo "Name is $name, and date is $(date)"
```

→ To print date.

TAKE USER INPUT

SCRIPT

```
#!/bin/bash
echo "enter the name : "
read username
echo " you entered $username
```

→ ./script.sh
↳ enter the name :
↳ jetha
↳ you entered jetha.

ARGUMENTS

```
./file.sh ____ ____
```

Argument 0

Argument 1

Argument 2

→ We can use these arguments in a script or Code.

↓ To print these argument

↓ let's make some changes in the script.

```
./file.sh ~~file1~~ ~~file2~~
```

only ($1) first argument will run → Because of no Access.

SCRIPT ..

```
echo "The character in
        $0 $1"
:wq
```

file   file1

OUTPUT

```
file file1
```

→ We can create multiple users using arguments.

Example :-

```
./create_user.sh
```

```
#!/bin/bash

if [ -z "$1" ]; then
echo "provide a username"
  exit 1
fi

 Sudo useradd $1
```

→ condition to check, as the username is provided or not.

→ username provided ! Exit Successfully

→ Add user.

↳ run →

```
./create_user Razzaq
```

↳ Argument passed "username provided"

CONDITIONAL STATEMENTS (if_else)

```
if [condition] ;
        then
        ≡
else ;
        ≡
```

Book - Content

In Bash, conditional expressions are used by the [[compound command and the [built-in commands to test file attributes and perform string and arithmetic comparisons.

Example

Script To check if-else .

```
#!/bin/bash
read -p "Enter number" number
if [[ $number == number/2 ]];
then echo "number is Even"
else
     echo "number is odd".
fi
```

accessing variable

condition

variable

```
chmod 755 numer.sh
```

↓ run the Script .

```
./number.sh
```

↳ Enter number

```
4 → number is Even.
5 → number is Odd.
```