Mastering Bash Scripting in Ubuntu

1. What is Bash?

- **Bash** = *Bourne Again Shell*, the default Ubuntu shell
- Acts as a command interpreter
- Executes both interactive commands and scripts
- Why bash scripting? 🔁 Automate repetitive tasks, manage servers, simplify workflows

---

2. Writing and Running a Bash Script

Basic Structure:

```
#!/bin/bash
# My first script
echo "Hello, World!"
```

Steps:

1. Save as hello.sh

Make it executable:

```
chmod +x hello.sh
```

2.

Run it:

./hello.sh

3.

Notes:

- #!/bin/bash is called a **shebang** – tells system which interpreter to use
- echo prints output
- Comments begin with #

---

3. Variables and Constants

Declaring variables:

```
name="Jehad"
age=23
```

- No space around =
- Access: $name, $age

```
echo "Name: $name"
```

Constants:

Use readonly keyword:

```
readonly PI=3.14
```

---

4. Reading User Input

```
echo "Enter your name:"
read username
```

```
echo "Welcome, $username!"
```

- Read multiple values:

```
read a b c
```

- With prompt on same line:

```
read -p "Enter your age: " age
```

- Silent input (e.g., password):

```
read -sp "Enter password: " pass
```

---

5. Conditionals and Logic

If-Else Structure:

```
if [ condition ]; then
  commands
elif [ condition ]; then
  commands
else
  commands
fi
```

Examples:

```
num=5
if [ $num -gt 10 ]; then
  echo "Greater than 10"
else
  echo "10 or less"
fi
```

Operators:

Type	Operators
Numeric	-eq, -ne, -lt, -le, -gt, -ge
String	=, !=, -z, -n
File	-e, -f, -d, -r, -w, -x

---

6. Loops in Bash

➤ For Loop:

```
for i in 1 2 3
do
  echo "Number $i"
done
```

➤ C-style For Loop:

```
for (( i=0; i<5; i++ ))
do
  echo $i
done
```

➤ While Loop:

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

➤ Until Loop:

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

---

7. Functions in Bash

```
greet() {
  echo "Hello, $1"
}

greet "Jehad"
```

- $1, $2, etc., are **positional parameters**
- You can return values using echo or global vars

---

8. Useful Bash Features

String operations:

```
str="BashScript"
echo ${str:0:4}   # Bash
echo ${#str}      # Length
```

Arrays:

```
fruits=("Apple" "Banana" "Cherry")
echo ${fruits[1]}        # Banana
echo ${fruits[@]}        # All
```

---

9. File and Directory Operations

```
mkdir mydir
cd mydir
touch file1.txt file2.txt
cp file1.txt backup.txt
mv file2.txt archive/
rm backup.txt
```

File existence check:

```
if [ -f "file1.txt" ]; then
  echo "File exists"
fi
```

---

10. Script Arguments and Exit Status

```
echo "Script name: $0"
echo "First argument: $1"
echo "All arguments: $@"

exit 0   # success
exit 1   # error

echo $?   # exit status of last command
```

---

11. Cron Jobs (Brief Intro)

- Use cron to schedule script execution

```
crontab -e

0 9 * * * /home/user/myscript.sh
```

This runs the script every day at 9:00 AM.

Practice Mini-Projects

1. Automated Backup Script

```
#!/bin/bash
src="/home/user/Documents"
dest="/home/user/Backup"
mkdir -p $dest
cp -r $src/* $dest/
```

```
echo "Backup completed at $(date)" >> $dest/backup.log
```

2. File Type Counter

```
#!/bin/bash
echo "Enter directory:"
read dir
cd "$dir"
echo "Files: $(ls -l | grep ^- | wc -l)"
echo "Directories: $(ls -l | grep ^d | wc -l)"
```

3. Disk Usage Monitor with Alert

```
#!/bin/bash
usage=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')
if [ $usage -gt 80 ]; then
  echo "Warning: Disk usage is above 80%!"
fi
```

Homework

- Create a script that:
    - Accepts a filename from the user
    - Checks if it exists
    - If it exists, shows line count, word count, and character count
- Write a script that loops through .txt files in a directory and counts how many lines contain a specific word
- Automate daily log file creation using a date-based filename