BASH SCRIPTING

Week 1
Bash Foundations

Write, run, and explain your first scripts

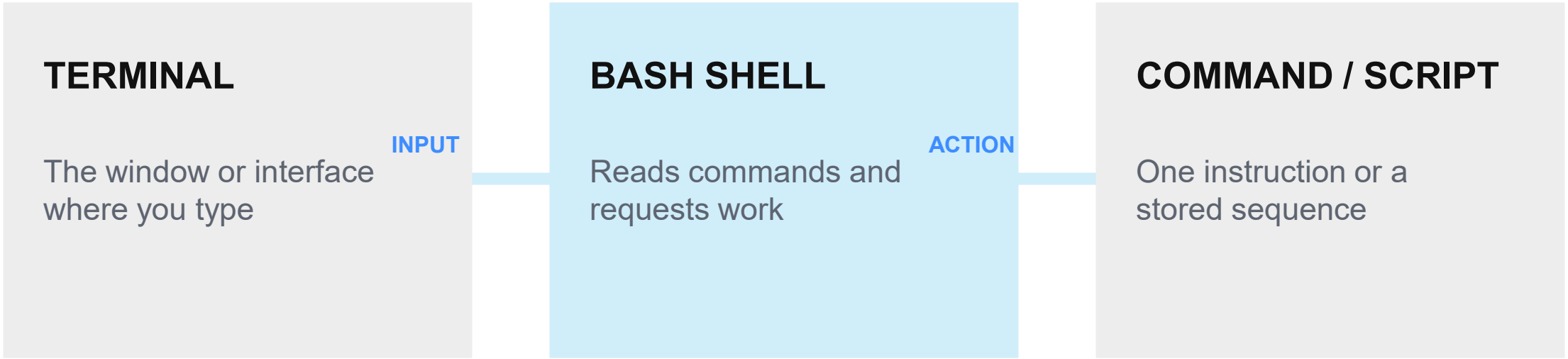
Two 90-minute lessons | Milestone M1: Script Operator

Instructor: Muhammad Khalid Khan

By the end of Week 1, you can prove six skills

01	Explain terminal, shell, Bash, command, and script
02	Identify the shebang, comments, commands, and output
03	Run a script with bash script.sh
04	Run a script directly with ./script.sh
05	Use exit status to recognize success or failure
06	Build and explain an environment snapshot

Terminal is the interface; Bash interprets commands



A terminal can host different shells. Bash is one shell and also a scripting language.

A script is a file that Bash reads from top to bottom

1

WRITE

Create a text file

2

SELECT

Choose Bash

3

EXECUTE

Run each command

4

OBSERVE

Read output + status

```
editor -> first-script.sh -> bash -> commands -> output + exit status
```

Four parts make a first script readable and runnable

```
#!/usr/bin/env bash

# Display identity information

printf '%s\n' 'System Identity'
whoami
hostname
pwd
```

01 **SHEBANG**

Selects Bash for direct execution.

02 **DESCRIPTION**

Explains the script's purpose.

03 **OUTPUT**

Gives the reader clear context.

04 **COMMANDS**

Perform the requested work.

The execution method determines which requirements apply

METHOD 1

```
bash script.sh
```

Bash is selected explicitly.
The file must be readable.
Execute permission is not required.

METHOD 2

```
./script.sh
```

The system reads the shebang.
The file must be executable.
./ means the current directory.

A .sh filename does not grant permission - chmod +x changes the permission bits.

Guided demo: predict every line before execution

```
#!/usr/bin/env bash

printf '%s\n' '=== My First Bash Script ==='
printf 'User: %s\n' "$(whoami)"
printf 'Hostname: %s\n' "$(hostname)"
printf 'Current directory: %s\n' "$(pwd)"
printf 'Date: %s\n' "$(date)"
```

RUN

```
bash first-script.sh

chmod +x first-
script.sh
./first-script.sh

echo $?
```

CHECK
Same output. Final status: 0.

Output, error, and status answer different questions

STANDARD OUTPUT

What information did the command produce?

```
pwd
/home/student
```

STANDARD ERROR

What diagnostic did the command report?

```
ls /missing
No such file...
```

EXIT STATUS

Did the command succeed or fail?

```
echo $?
0 or nonzero
```

Check $? immediately - the next command replaces it.

&& continues on success; || responds to failure

SUCCESS PATH

```
command1 && command2
```

command1 returns 0

then command2 runs

FAILURE PATH

```
command1 || command2
```

command1 returns nonzero

then command2 runs

```
pwd && echo 'ready'     |     ls /missing || echo 'check failed'
```

A useful report combines commands with clear labels

```
#!/usr/bin/env bash

printf '%s\n' 'Linux Environment Snapshot'
printf 'Date: %s\n' "$(date)"
printf 'User: %s\n' "$(whoami)"
printf 'Hostname: %s\n' "$(hostname)"
printf 'Kernel: %s\n' "$(uname -r)"
uptime
free -h
df -h /
```

REPORT SECTIONS

01 Identity

02 Operating system

03 Memory

04 Root filesystem

05 Completion message

Read-only by design: no sudo, no configuration changes.

Two labs turn understanding into evidence

LAB 1

My First Script

Create identity output
Run through Bash
Run directly
Observe permission failure
Explain every line

LAB 2

Environment Snapshot

Check syntax
Report system information
Use readable sections
Test two execution methods
Record final exit status

Required proof: script + commands + output + exit status + explanation

Milestone M1: create, execute, and explain independently

SCRIPT OPERATOR

You are ready when you can build a script, run it two ways, check status, and explain every line.

01 **STRUCTURE**
Shebang + description + commands

02 **EXECUTION**
bash file and ./file both work

03 **EVIDENCE**
Output + error + exit status

NEXT: Week 2 adds variables, quoting, user input, and positional arguments.