Linux Shell Scripting

Contents

- Introduction to Shell Scripting
- Basic Shell Script Structure
- Shell Variables and Input/Output
- Conditional Statements (if-else)
- Loops in Shell Scripts
- Functions in Shell Scripts
- Real-World Use Case: Automation Script

Introduction to Shell Scripting

Introduction to Shell Scripting

What is Shell Scripting ?

Introduction to Shell Scripting

What is Shell Scripting ?

Definition

A **shell-script** is a **file** containing a series of commands that are executed by the shell

- Shell: The command interpreter in Linux

Introduction to Shell Scripting

What is Shell Scripting ?

Definition

A **shell-script** is a **file** containing a series of commands that are executed by the shell

- Shell: The command interpreter in Linux

Use Cases

Automating repetitive tasks, system monitoring, batch processing, data manipulation, backups, etc.

Introduction to Shell Scripting

What is Shell Scripting ?

Linux Shells

Shell	User-Friendliness	Scripting Power	Customization	Performance	Portability
Bash	High	High	High	Moderate	High
sh	Low	Moderate	Low	High	Very High
Zsh	Very High	High	Very High	Moderate	Moderate
Ksh	Moderate	High	Moderate	High	High
Csh	Moderate	Moderate	Low	Moderate	Low
Tcsh	High	Moderate	Low	Moderate	Low
Dash	Low	Low	Low	Very High	Very High
Fish	Very High	Moderate	High	Moderate	Low

Introduction to Shell Scripting

What is Shell Scripting ?

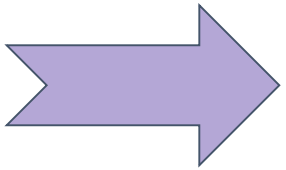
Shell	User-Friendliness	Scripting Power	Customization	Performance	Portability
Bash	High	High	High	Moderate	High
sh	Low	Moderate	Low	High	Very High
Zsh					
Csh					
Tcsh	High	Moderate	Low	Moderate	Low
Dash	Low	Low	Low	Very High	Very High
Fish	Very High	Moderate	High	Moderate	Low

Linux Shells

- **Bash (Bourne Again Shell)**: The most common shell in Linux. ***We'll focus on Bash for this session.***

Basic Shell Script Structure

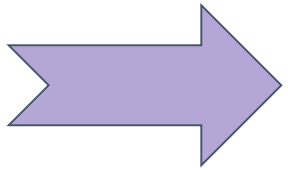
Creating and Running Shell Scripts



Create a Shell Script File

Basic Shell Script Structure

Creating and Running Shell Scripts



Create a Shell Script File

$ vim hello.sh

Basic Shell Script Structure

Creating and Running Shell Scripts

Create a Shell Script File

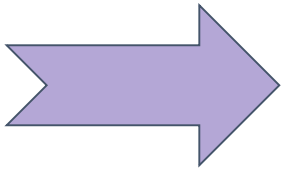
$ vim hello.sh

```
echo "Hello, World"
```

hello.sh

Basic Shell Script Structure

Creating and Running Shell Scripts

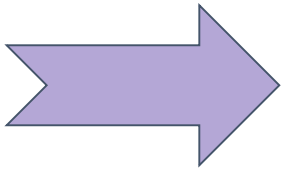


Add the Shebang (#!/bin/bash)

```
#!/bin/bash
echo “Hello, World”
```

Basic Shell Script Structure

Creating and Running Shell Scripts



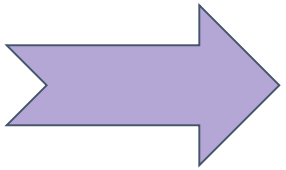
Add the Shebang (#!/bin/bash)

```
#!/bin/bash
echo "Hello, World"
```

The shebang (#!) tells the system which interpreter to use.

Basic Shell Script Structure

Creating and Running Shell Scripts



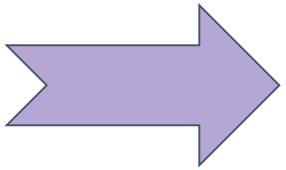
Make the Script Executable

```
#!/bin/bash
echo "Hello, World"
```

$ chmod +x hello.sh

Basic Shell Script Structure

Creating and Running Shell Scripts



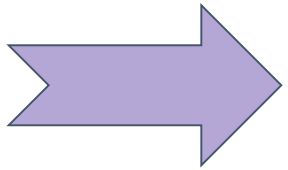
Run the Script

```
#!/bin/bash
echo "Hello, World"
```

$ ./hello.sh

Basic Shell Script Structure

Creating and Running Shell Scripts

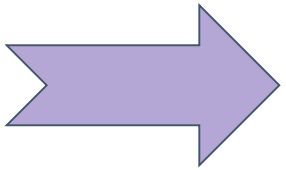


Example-1

```
[root@localhost Test2]# vi hello.sh
[root@localhost Test2]# cat hello.sh
#!/bin/bash
echo "Hello, World"
[root@localhost Test2]# chmod +x hello.sh
[root@localhost Test2]# ls
hello.sh
[root@localhost Test2]# ./hello.sh
Hello, World
[root@localhost Test2]#
```

Basic Shell Script Structure

Creating and Running Shell Scripts

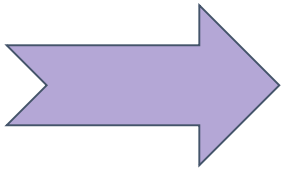


Example-2: Print the Current Date

```
#!/bin/bash
echo "Date is: $(date)"
```

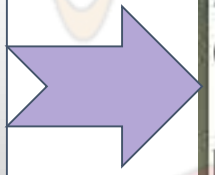
Basic Shell Script Structure

Creating and Running Shell Scripts



Example-2: Print the Current Date

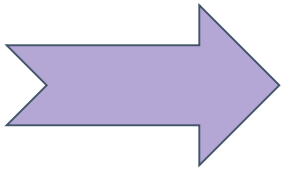
```
#!/bin/bash
echo "Date is: $(date)"
```



```
[root@localhost Test2]# cat date.sh
#!/bin/bash
echo "Date is: $(date)"
[root@localhost Test2]# ./date.sh
Date is: Fri Oct  4 07:36:19 PDT 2024
[root@localhost Test2]#
```

Basic Shell Script Structure

Creating and Running Shell Scripts

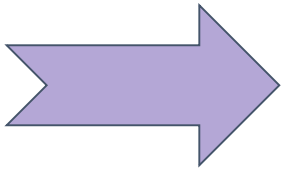


Example-3: Addition of two numbers

```
#!/bin/bash
echo "5 + 2 = $((5 + 2))"
```

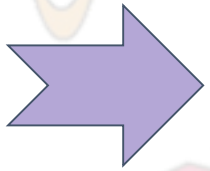
Basic Shell Script Structure

Creating and Running Shell Scripts



Example-3: Addition of two numbers

```
#!/bin/bash
echo "5 + 2 = $((5 + 2))"
```



```
[root@localhost Test2]# ls
add.sh  date.sh  hello.sh
[root@localhost Test2]# ./add.sh
5 + 2 = 7
[root@localhost Test2]#
```

Shell Variables and Input/Output

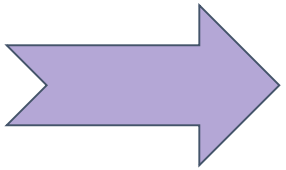
Shell Variables and Input/Output

Variables in Shell Scripts

Variables are used to **store** information (like numbers or strings) that can be used later in the script.

Shell Variables and Input/Output

Variables in Shell Scripts

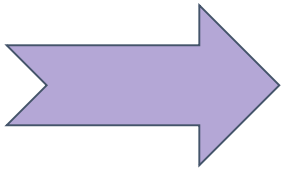


Example-1: Simple Variable

```
#!/bin/bash
name="Pitter"
echo "Hello, $name!"
```

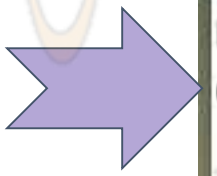
Shell Variables and Input/Output

Variables in Shell Scripts



Example-1: Simple Variable

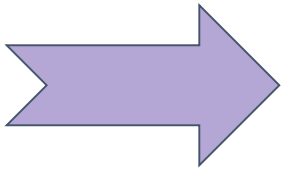
```
#!/bin/bash
name="Pitter"
echo "Hello, $name!"
```



```
[root@localhost Test2]# cat name.sh
#!/bin/bash
name="Pitter"
echo "Hello, $name!"
[root@localhost Test2]# ./name.sh
Hello, Pitter!
[root@localhost Test2]#
```

Shell Variables and Input/Output

Variables in Shell Scripts

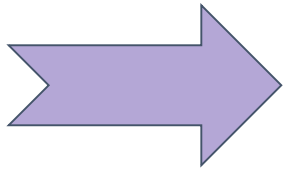


Example-2: Using Command Substitution

```
#!/bin/bash
current_time=$(date +%T)
echo "The current time is
$current_time"
```

Shell Variables and Input/Output

Variables in Shell Scripts



Example-2: Using Command Substitution

```
#!/bin/bash
current_time=$(date +%T)
echo "The current time is
$current_time"
```

```
[root@localhost Test2]# cat substitute.sh
#!/bin/bash
current_time=$(date +%T)
echo "The current time is $current_time"
[root@localhost Test2]# ./substitute.sh
The current time is 07:58:11
[root@localhost Test2]#
```