EPIC
Institute of Technology
Powered by <epam>

# Linux CLI commands and Bash Scripting

Ilya Slimianiou

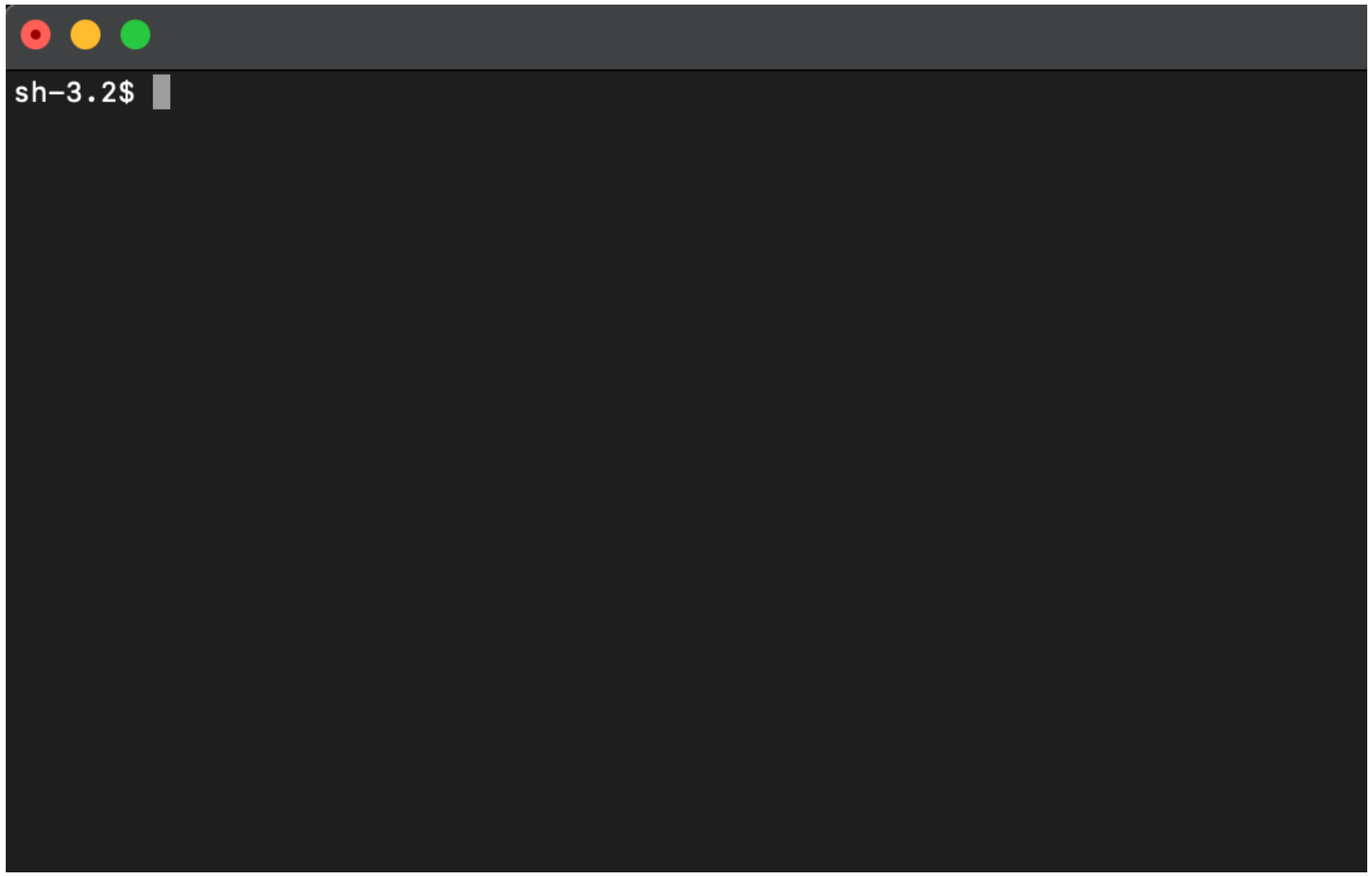
March–June 2024

EPIC
Institute of Technology
Powered by <epam>

# Contents

EPIC
Institute of Technology
Powered by <epam>

# Shell




**Thompson shell**

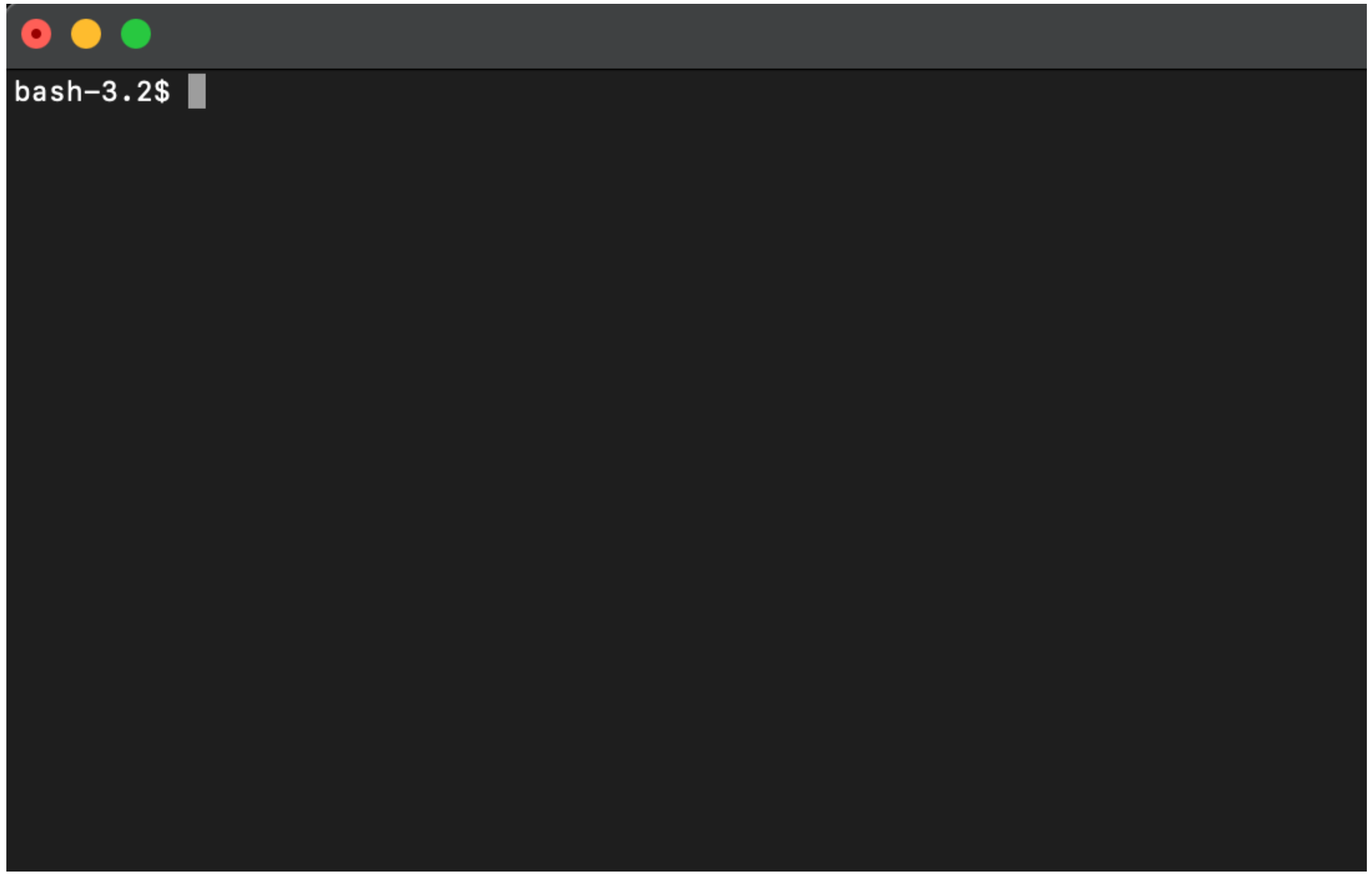
**Bourne Shell**

**C Shell**

**Korn Shell**

EPIC
Institute of Technology
Powered by <epam>

# Bash




**Bourne-Again SHell**

**Created by Brian Fox in 1989**

EPIC
Institute of Technology
Powered by <epam>

# What you need to know to build a script?

Use shebang **#!**

Make your script executable

Type **./** in front of you script file name to execute

EPIC
Institute of Technology
Powered by <epam>

# Shebang

# is called "sharp"

! is called "bang"

the late 1970s

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

EPIC Institute of Technology
Powered by <epam>

# Variables

## Local Variables:

```
function myFunction {
# Define a local variable
local local_var="Available Only in Function"
echo "Inside function: $shell_var, $local_var"
}
```

## Environment Variables:

```
export VARIABLE_NAME="value”
```

## Shell Variables:

```
DAY="Tuesday"
```

EPIC Institute of Technology
Powered by <epam>

# Parameters

**Shell Parameters**

| Parameters | Function |
|---|---|
| $1-$9 | Represent positional parameters for arguments one to nine |
| ${10}-${n} | Represent positional parameters for arguments after nine |
| $0 | Represent name of the script |
| $* | Represent all the arguments as a single string |
| $@ | Same as $*, but differ when enclosed in (") |
| $# | Represent total number of arguments |
| $$ | PID of the script |
| $? | Represent last return code |

EPIC
Institute of Technology
Powered by <epam>

# Input Variables

```
#!/bin/bash
echo "Script Name: $0"
echo "First Argument: $1"
echo "Second Argument: $2"
```

Running this script as `./script.sh AWS EC2` would output:

```
Script Name: ./script.sh
First Argument: AWS
Second Argument: EC2
```

EPIC
Institute of Technology
Powered by <epam>

# Status codes

```
# Example: Checking the status code of a command
ls /nonexistent_directory
echo $? # This will print the status code of the 'ls' comma
nd. Since the directory does not exist, it is likely to be
non-zero.
```

```
# Example: Silencing stderr
grep 'text' /nonexistent_file 2> /dev/null

# Example: Silencing both stdout and stderr
find / -name 'myfile' > /dev/null 2>&1
```

EPIC
Institute of Technology
Powered by <epam>

# User Input

```
#!/bin/bash
read -p "Enter your name: " name
echo "Welcome, $name!"
```

EPIC Institute of Technology
Powered by <epam>

# Piping and Command Substitution

- **Piping Example:**

```
echo "This is a test" | wc -w
```

This counts the words in the string `This is a test`, outputting `4`.

- **Command Substitution Example:**

```
#!/bin/bash
files=$(ls)
echo "Files in directory: $files"
```

EPIC
Institute of Technology
Powered by <epam>

# Arrays

### Define a bash array

You have two ways to create a new array. The first one is to `declare` command with `-a` option to define an array.

```
$ declare -a test_array
```

Or, you can simply create Array by assigning elements.

```
$ test_array=("one" "two" "three")
```

```
test_array+=("four" "five")
```

```
$ echo ${test_array[0]}
one
```

EPIC
Institute of Technology
Powered by <epam>

# If else

```
#!/bin/bash
if [ "$1" == "hello" ]; then
  echo "Hello there!"
else
  echo "Who are you?"
fi
```

EPIC
Institute of Technology
Powered by <epam>

# Comparison

- Equal: `eq`
- Not equal: `ne`
- Less than or equal: `le`
- Less than: `lt`
- Greater than or equal: `ge`
- Greater than: `gt`
- Is null: `z`

- Equal: `==`
- Not equal: `!=`

```
if [ "$foo" == "$bar" ]
```

EPIC
Institute of Technology
Powered by <epam>

# For loops

```
#!/bin/bash
s=("football" "basketball" "volleyball")
for n in ${s[@]}
do
  echo $n
done
```

EPIC Institute of Technology
Powered by <epam>

# Functions

```
#!/bin/bash

# Define a shell variable
shell_var="Available Everywhere in Shell"

function myFunction {
    # Define a local variable
    local local_var="Available Only in Function"
    echo "Inside function: $shell_var, $local_var"
}

# Call the function
myFunction
```

EPIC Institute of Technology
Powered by <epam>

# Summary

Bash scripting is a powerful tool primarily used for automating tasks and managing system operations on Unix-like operating systems

Main benefit is Bash's integration with the Linux environment and its ability to directly use Unix command line utilities