What is Batch Script?

- ❖ Batch Script is a scripting language that simplifies the process of executing repetitive tasks or commands on Windows machines. It allows users to write a series of commands in a plain text file with a .bat or .cmd extension, which can then be executed as a single script.

Basic Commands in Batch Script:

Echo: Displays messages on the screen.

Rem: Adds comments in the script (remarks).

Set: Assigns a value to a variable.

If: Executes a command conditionally based on the result of a comparison.

For: Loops through a set of items and executes a command for each item.

Goto: Redirects the execution flow to a specific label within the script.

Call: Calls another batch file from within the current script.

Pause: Pauses the execution of the script until a key is pressed.

Exit: Exits the script.

- ❖ These are some of the basic commands used in Batch Scripting. With these commands, you can perform a wide range of tasks from simple file manipulation to complex system administration tasks on Windows.

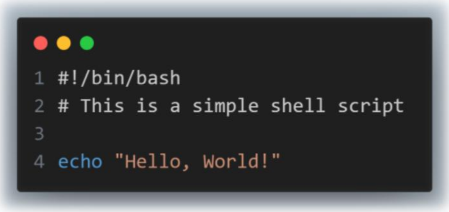
What is Shell Scripting?

- Shell scripting is the process of creating and running scripts written in a shell language.
- The shell is a command-line interpreter that provides a user interface for accessing the operating system's services.
- The most common shell on Unix-like systems is Bash (Bourne Again Shell), although other shells like Zsh (Z shell) and Ksh (KornShell) also exist.

Here's a brief overview of how you can create and execute shell scripts on Ubuntu:

Create a Shell Script:

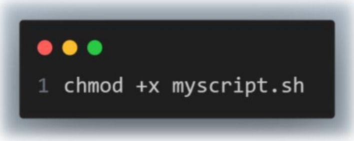
- You can create a shell script using any text editor. Here's a basic example:



```
#!/bin/bash
# This is a simple shell script

echo "Hello, World!"
```

Make the Script Executable:

- Before you can run the script, you need to make it executable. You can do this using the chmod command:



```
chmod +x myscript.sh
```

Run the Script:

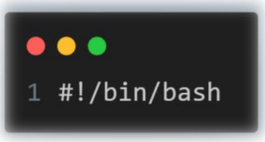
- You can run the script by typing its name preceded by ./ in the terminal.

Error Handling:

- You can use the (**set -e**) option to automatically exit the script if any command fails, ensuring robust error handling

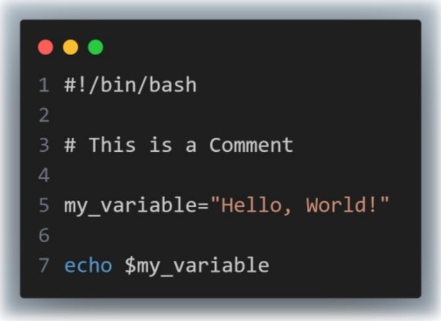
Shebang (#!):

- The shebang is the first line of a script and specifies the interpreter to use. For Bash scripts, the shebang line typically looks like this:



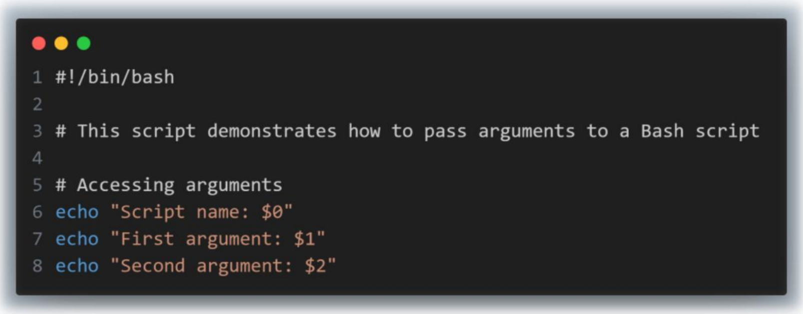
```
#!/bin/bash
```

using Variables and Comments



```bash
#!/bin/bash

# This is a Comment

my_variable="Hello, World!"

echo $my_variable
```

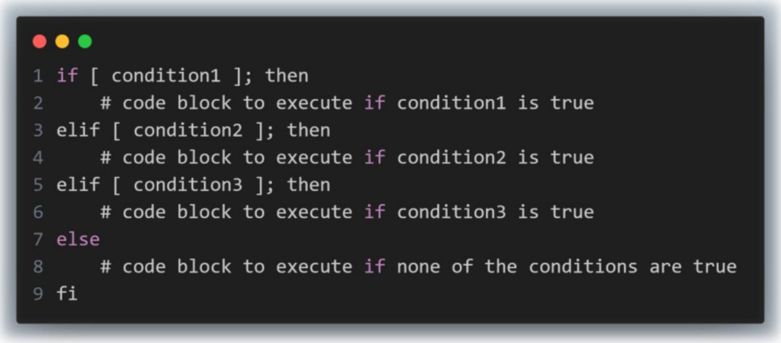
Pass Arguments to a Bash-Script



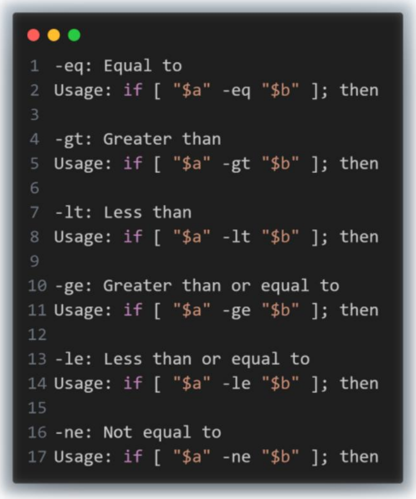
```bash
#!/bin/bash

# This script demonstrates how to pass arguments to a Bash script

# Accessing arguments
echo "Script name: $0"
echo "First argument: $1"
echo "Second argument: $2"
```

- You can pass as many arguments as needed, and they will be accessible using $1, $2, $3, and so on, up to $9. Beyond $9, you need to use curly braces ${10}, ${11}, and so forth.
- If you need to access all the arguments as a single string, you can use the special variable $@ or $*.

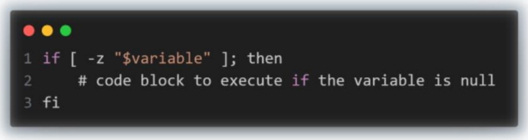
If Statement (If then , If then else, If elif else)



```
if [ condition1 ]; then
    # code block to execute if condition1 is true
elif [ condition2 ]; then
    # code block to execute if condition2 is true
elif [ condition3 ]; then
    # code block to execute if condition3 is true
else
    # code block to execute if none of the conditions are true
fi
```



```
-eq: Equal to
Usage: if [ "$a" -eq "$b" ]; then

-gt: Greater than
Usage: if [ "$a" -gt "$b" ]; then

-lt: Less than
Usage: if [ "$a" -lt "$b" ]; then

-ge: Greater than or equal to
Usage: if [ "$a" -ge "$b" ]; then

-le: Less than or equal to
Usage: if [ "$a" -le "$b" ]; then

-ne: Not equal to
Usage: if [ "$a" -ne "$b" ]; then
```

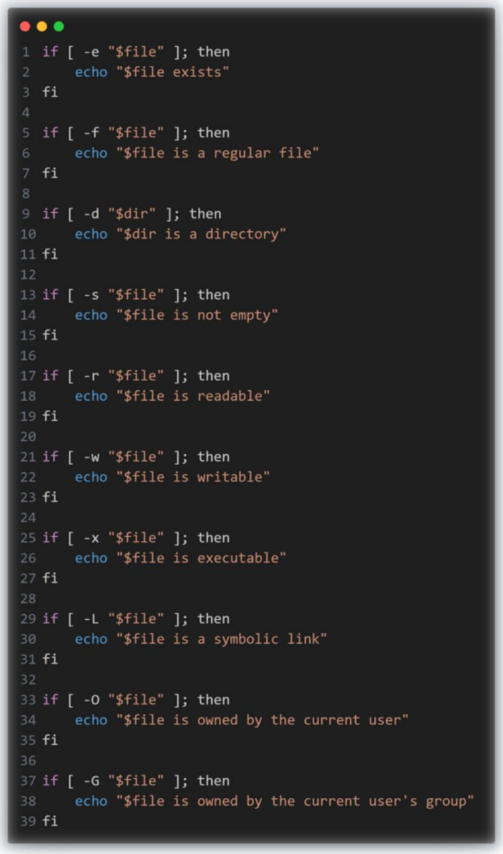


```
if [ -z "$variable" ]; then
    # code block to execute if the variable is null
fi
```

- Always wrap variables and expressions inside square brackets [ ] when using them in conditions.
- Use the appropriate comparison operators **(-eq, -gt, -lt, -ge, -le, -ne)** for numerical comparisons and **(==)** for string comparisons within square brackets.
- Always terminate each if, elif, else block with **(fi)** to signify the end of the block.

File test operators

-e: Checks if a file exists.
-f: Checks if a file exists and is a regular file
(not a directory or device file).
-d: Checks if a file exists and is a directory.
-s: Checks if a file exists and is not empty
(has a size greater than zero).
-r: Checks if a file exists and is readable.
-w: Checks if a file exists and is writable.
-x: Checks if a file exists and is executable.
-x: Checks if a file exists and is executable.
-x: Checks if a file exists and is executable.
-G: Checks if a file exists and is owned by
the current user's group.



```
if [ -e "$file" ]; then
    echo "$file exists"
fi

if [ -f "$file" ]; then
    echo "$file is a regular file"
fi

if [ -d "$dir" ]; then
    echo "$dir is a directory"
fi

if [ -s "$file" ]; then
    echo "$file is not empty"
fi

if [ -r "$file" ]; then
    echo "$file is readable"
fi

if [ -w "$file" ]; then
    echo "$file is writable"
fi

if [ -x "$file" ]; then
    echo "$file is executable"
fi

if [ -L "$file" ]; then
    echo "$file is a symbolic link"
fi

if [ -O "$file" ]; then
    echo "$file is owned by the current user"
fi

if [ -G "$file" ]; then
    echo "$file is owned by the current user's group"
fi
```

File Operations

❖ Here are some commonly used commands for moving, copying, and deleting files in Unix/Linux environments, along with a few additional commands for file management

mv (Move):

- The mv command is used to move files or directories from one location to another.
- It can also be used to rename files

cp (Copy):

- The cp command is used to copy files or directories.
- To copy directories and their contents recursively, use the -r or -R option:

rm (Remove):

- The rm command is used to delete files or directories.

mkdir :

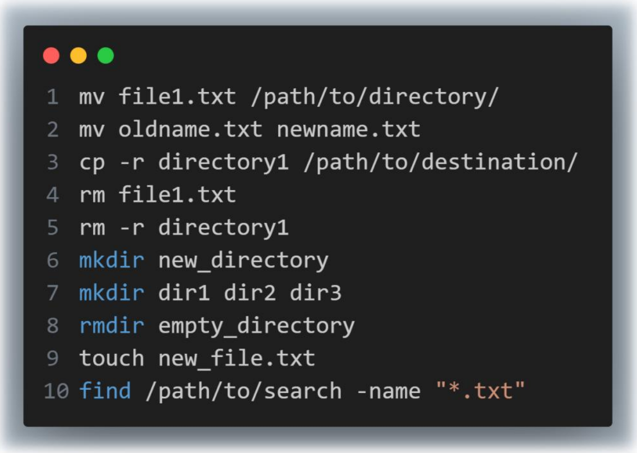
- The mkdir command is used to create directories.
- To create multiple directories at once, you can provide multiple directory names:

rmdir (Remove Directory):

- The rmdir command is used to delete empty directories.

find:

- The find command is used to search for files and directories within a directory hierarchy.



```
mv file1.txt /path/to/directory/
mv oldname.txt newname.txt
cp -r directory1 /path/to/destination/
rm file1.txt
rm -r directory1
mkdir new_directory
mkdir dir1 dir2 dir3
rmdir empty_directory
touch new_file.txt
find /path/to/search -name "*.txt"
```