ANJALI KUMARI

DAY7

LINUX SCRIPTING SHELL

ASSIGNMENT1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Below is a simple shell script that checks if a specific file (e.g., myfile.txt) exists in

the current directory and prints the appropriate message:

```
#!/bin/bash
# Check if the file exists
if [ -f "myfile.txt" ]; then
echo "File exists"
else
echo "File not found"
fi
```

- #!/bin/bash: This line specifies the shell to be used to execute the script, in this case, Bash.
- [-f "myfile.txt"]: This is the condition that checks if the file myfile.txt exists in the current directory. The -f flag checks if the file exists and is a regular file.
- echo "File exists": If the file exists, this command prints "File exists" to the

standard output.

- echo "File not found": If the file does not exist, this command prints "File not found" to the standard output.

Assignment 2: Write a script that reads numbers from the user until they enter '0'.The script should also print whether each number is odd or even.

Here's a simple shell script that reads numbers from the user until they enter '0' and prints whether each number is odd or even:

```
#!/bin/bash
echo "Enter numbers (enter '0' to exit):"
while true; do
read -p "Enter a number: " number
# Check if the input is '0'
if [ "$number" -eq 0 ]; then
echo "Exiting..."
break
fi
# Check if the number is odd or even
if [ "$((number % 2))" -eq 0 ]; then
echo "$number is even"
else
echo "$number is odd"
```

```
fi
done
```

● #!/bin/bash: Specifies the shell to be used to execute the script.

● echo "Enter numbers (enter '0' to exit):": Prints a message prompting the user to enter numbers.

● while true; do: Starts an infinite loop.

● read -p "Enter a number: " number: Prompts the user to enter a number and stores the input in the variable number.

● if ["$number" -eq 0]; then: Checks if the input is '0'. If it is, the script prints "Exiting..." and breaks out of the loop.

● if ["$((number % 2))" -eq 0]; then: Checks if the number is even by calculating the remainder when divided by 2. If the remainder is 0, the number is even; otherwise, it's odd.

● Inside the loop, the script continuously prompts the user for numbers until '0' is entered, printing whether each number is odd or even.

You can save this script to a file (e.g., check_odd_even.sh), make it executable with the command chmod +x check_odd_even.sh, and then run it with ./check_odd_even.sh

Assignment 3: Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.

Below is a shell script that defines a function count_lines_in_file which takes a filename as an argument and prints the number of lines in the file. It then calls this function with different filenames:

```
#!/bin/bash
# Function to count lines in a file
count_lines_in_file() {
local filename="$1"
local num_lines=$(wc -l < "$filename")
echo "Number of lines in $filename: $num_lines"
}
# Call the function with different filenames
count_lines_in_file "file1.txt"
count_lines_in_file "file2.txt"
count_lines_in_file "file3.txt"
```

- #!/bin/bash: Specifies the shell to be used to execute the script.
- count_lines_in_file() { ... }: Defines a function named count_lines_in_file which takes a filename as an argument.
- local filename="$1": Assigns the value of the first argument (the filename) to the variable filename.
- local num_lines=$(wc -l < "$filename"): Uses the wc command to count the number of lines in the file specified by the filename. The < symbol is used for input redirection. The output of wc -l is stored in the variable num_lines.

● echo "Number of lines in $filename: $num_lines": Prints the filename

and the number of lines in the file.

● The script then calls the count_lines_in_file function with different

filenames: "file1.txt", "file2.txt", and "file3.txt".

Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

Below is a shell script that creates a directory named TestDir and inside it, creates

ten files named File1.txt, File2.txt, ..., File10.txt. Each file contains its

filename as its content:

```
#!/bin/bash
# Create the directory TestDir if it doesn't exist
mkdir -p TestDir
# Navigate to the TestDir directory
cd TestDir || exit
# Create ten files named File1.txt, File2.txt, ..., File10.txt
for ((i = 1; i <= 10; i++)); do
filename="File$i.txt"
echo "$filename" > "$filename"
done
echo "Files created successfully."
```

● #!/bin/bash: Specifies the shell to be used to execute the script.

● mkdir -p TestDir: Creates the directory TestDir if it doesn't already exist. The -p option ensures that the command doesn't produce an error if the directory already exists.

● cd TestDir || exit: Navigates into the TestDir directory. If for some reason navigation fails, the script exits.

● for ((i = 1; i <= 10; i++)); do: Starts a loop to create ten files.

● filename="File$i.txt": Constructs the filename for each iteration of the loop (e.g., File1.txt, File2.txt, ..., File10.txt).

● echo "$filename" > "$filename": Writes the filename (e.g., "File1.txt") into the corresponding file.

● echo "Files created successfully.": Prints a message indicating that the files have been created successfully