Linux and Shell Scripting

Assignment 01

Assignment 1: 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".

Answer:

- First open the VI editor.
- Then create VI file as “VI filename.sh” it will open in VI
- Type I and write the above code to save, press Esc
- Press Ctrl+: wq if will exit the VI editor and came to execute step
- To execute type “bash filename.sh

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

- Save this script in a file as **check_file.sh**, then make it executable using the following command:

```
chmod +x check_file.sh
```

- Now you can run the script:

```
Bash check_file.sh
```

- If ‘myfile.txt’ exists in the current directory, it will print "File exists". Otherwise, it will print "File not found".

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.

Answer:

- Open a terminal.
- Create a new shell script file.
- Open the file with a text editor and add the following code:

```
!/bin/bash
echo "Enter numbers (enter 0 to exit):"
while true; do
   read -p "Enter a number: " num
```

```
    if [ $num -eq 0 ]; then
        echo "Exiting..."
        break
    fi

    if [ $((num % 2)) -eq 0 ]; then
        echo "$num is even."
    else
        echo "$num is odd."
    fi
done
```

- Save the script in a file, let's say odd_even_checker.sh, then make it executable using the following command:

```
chmod +x odd_even_checker.sh
```

- Now you can run the script:

```
./odd_even_checker.sh
```

- example output of running the script:

```
Enter numbers (enter 0 to exit):
Enter a number: 5
5 is odd.
Enter a number: 8
8 is even.
Enter a number: 0
Exiting...
```

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.

Answer:

- Open a terminal.
- Create a new shell script file. You can name it `count_line.sh`.
- Open the file with a text editor and add the following code:

```
#!/bin/bash
countLine() {
    local file="$1"
    if [ -f "$file" ]; then
```

```
        local line=$(wc -l < "$file")
        echo "This file has $line lines."
    else
        echo "This file does not exist."
    fi
}

c_line "file1.txt"
c_line "file2.txt"
```

output:

This files has 1 lines.

This files has 2 lines

- Save the file and close the editor.
- Make the script executable by running the following command in the terminal:
  chmod +x count_line.sh
- Run the script by typing:
  bash count_line.sh

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").

Answer:

- To create a script that creates a directory named `TestDir` and inside it, creates ten files named `File1.txt`, `File2.txt`, ... `File10.txt`, with each file containing its filename as its content

```
#!/bin/bash
# Create directory TestDir if it doesn't exist
mkdir -p TestDir

# Loop to create ten files with the required content
for ((i=1; i<=10; i++)); do
    filename="File${i}.txt"
    echo "$filename" > "TestDir/$filename"
done

echo "Files created successfully in TestDir."
```

- Save the file and close the editor.
- Make the script executable by running the following command in the terminal:
  chmod +x create_files.sh

- Run the script by typing: **bash create_files.sh**

Assignment 5: Modify the script to handle errors, such as the directory already existing or lacking permissions to create files.

Add a debugging mode that prints additional information when enabled.

Answer:

- Open the terminal.
- Edit your existing script `create_files.sh` or create a new one with the same name.
- Open the file with a text editor and update the code as follows:

```
#!/bin/bash

# Debugging mode function
enable_debug() {
   debug_mode=true
   echo "Debug mode enabled."
}

# Function to create files and handle errors
create_files() {
   local dir_name="TestDir"
   local filename

   # Check if directory already exists
   if [ -d "$dir_name" ]; then
      echo "Error: Directory '$dir_name' already exists."
      return 1
   fi

   # Create the directory
   mkdir -p "$dir_name"
   if [ $? -ne 0 ]; then
      echo "Error: Unable to create directory '$dir_name'."
      return 1
   fi

   # Change into the TestDir directory
   cd "$dir_name" || return 1

   # Loop to create files File1.txt through File10.txt
   for (( i = 1; i <= 10; i++ )); do
      filename="File${i}.txt"
```

```
        echo "$filename" > "$filename"
        if [ $? -ne 0 ]; then
            echo "Error: Unable to create file '$filename'."
            return 1
        fi
    done

    # Move back to the original directory
    cd ..
    return 0
}

# Main script
if [ "$1" = "--debug" ]; then
    enable_debug
fi

create_files
if [ $? -eq 0 ]; then
    echo "Files created successfully."
else
    echo "Failed to create files."
Fi
```

- Save the file and close the editor.
- Make the script executable by running the following command in the terminal: **chmod +x create_files.sh**
- Run the script by typing: bash **create_files.sh**
- To run the script with debugging mode enabled, type: **bash create_files.sh debug**

Assignment 6: Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line.

Data Processing with sed

Answer:

- Open the terminal.
- Create a sample log file named `sample.log` with some log entries. For example:

```
2024-05-20 08:35:01 INFO: Starting application
2024-05-20 08:35:10 ERROR: Database connection failed
```

2024-05-20 08:36:15 ERROR: Out of memory
2024-05-20 08:37:00 DEBUG: Received request from 192.168.1.100
2024-05-20 08:37:05 ERROR: Disk full

- Create a new shell script file named `process_logs.sh`.
- Open the file with a text editor and add the following code:

```
#!/bin/bash

# Define the log file
logfile="sample.log"

# Check if log file exists
if [ ! -f "$logfile" ]; then
   echo "Error: Log file '$logfile' not found."
   exit 1
fi

# Check if log file is readable
if [ ! -r "$logfile" ]; then
   echo "Error: Log file '$logfile' is not readable."
   exit 1
fi

# Extract lines containing "ERROR" using grep, and print date, time, and error message using awk
echo "Using grep and awk:"
grep "ERROR" "$logfile" | awk -F ': ' '{print $1, $2}'

# Additional processing example with sed: extract only the error message
echo "Additional processing with sed:"
grep "ERROR" "$logfile" | sed 's/^[^:]*:[^:]*: //'
```

- Save the file and close the editor.
- Make the script executable by running the following command in the terminal: **chmod +x process_logs.sh**
- Run the script by typing: **bash process_logs.sh**

Assignment 7: Create a script that takes a text file and replaces all occurrences of "old_text" with "new_text". Use sed to perform this operation and output the result to a new file.

Answer:

- Open a terminal.
- Create a new shell script file named `**replace_text.sh**`.
- Open the file with a text editor and add the following code:

```
#!/bin/bash

# Check if the correct number of arguments are passed
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 <input_file> <old_text> <new_text>"
    exit 1
fi

input_file="$1"
old_text="$2"
new_text="$3"

# Check if the input file exists
if [ ! -f "$input_file" ]; then
    echo "Error: Input file '$input_file' not found."
    exit 1
fi

# Define the output file name
output_file="${input_file%.txt}_replaced.txt"

# Perform the replacement using sed and write to the output file
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

echo "Replacement complete. Result written to $output_file."
```

- Save the file and close the editor.
- Make the script executable by running the following command in the terminal: **chmod +x replace_text.sh**
- Run the script by typing (replace `input.txt`, `old_text`, and `new_text` with your actual file and text): **bash replace_text.sh input.txt old_text new_text**

Hariharane01@gmail.com