Day 7 Assignment

prajwalhonashetti143@gmail.com

1. **ASSIGNMENT 1: Ensure the script check if the specific file(e. g myfile.txt) exists in the current directory. if it exists, print ”File exists”, otherwise print “file not found”**

Instruction:

- 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

File=”name.txt”

if [-e “$File”]: then

echo “File Exists”

else

echo “file not exists”

fi

Explanation: -

`#!/bin/bash`: This is the shebang line that tells the system to use the Bash shell to interpret the script.

- `FILE="myfile.txt"`: This line assigns the name of the file to check to the variable `

FILE`. - `if [-e "$FILE"]; then`: This checks if the file exists in the current directory. The `-e` option is used to check if a file exists. –

`echo "File exists"`: This line is executed if the file exists. –

**`echo "File not found**"`: This line is executed if the file does not exist.

Day 7 Assignment

prajwalhonashetti143@gmail.com

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

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

#!/bin/bash

Echo "Enter the number:"

read num

while ["$num" -ne 0] ; do

if [$((num % 2)) -eq 0]; then

echo "$num is even"

else

echo "$num is odd"

fi

echo "enter the number"

read num

done

echo "exit"

output:

enter the number : 20

20 is even

4. Save the file and close the editor.

5. Make the script executable by running the following command in the terminal:

``` **chmod +x filename.sh** ```

6. Run the script by typing: ``` ./filename.sh
```

prajwalhonashetti143@gmail.com

Explanation:

Infinite loop: The while : loop keeps the script running indefinitely until the user decides to stop by entering '0'.

Prompt and read input: The read -p command prompts the user and reads the input into the variable num.

Input validation: The script uses a regular expression to check if the input is a valid integer.

Case statement: The script uses a case statement to handle different inputs. If the input is '0', it prints an exit message and breaks the loop. For any other input, it checks if the number is even or odd and prints the corresponding message.

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.**

   1.Open a terminal.
   2. Create a new shell script file. You can name it **`c_line.sh`.**
   3. Open the file with a text editor and add the following code:

```
#!/bin/bash

c_line() {
 local file=$1

if [ -f “$file” ]; then
        local line=$(wc -l <$file” )
        echo “this files has $line lines.”
else
        echo “this file does not exists”
fi
}
c_line ”file1.txt”
c_line “file2.txt”
```

output:

This files has 1 lines.

This files has 2 lines.

prajwalhonashetti143@gmail.com

4. Save the file and close the editor.

5. Make the script executable by running the following command in the terminal: `

`` **chmod +x c_line.sh** ```

6. Run the script by typing: ``` **./c_line.sh** ```

Explanation: -

`#!/bin/bash`: This is the shebang line that tells the system to use the Bash shell to interpret the script.

- **`count_lines() { ... }`:** This function takes a filename as an argument and prints the number of lines in the file. –

`local filename="$1"`: This line assigns the first argument passed to the function to the variable `filename`. –

`if [ -f "$filename" ]; then`: This line checks if the file exists.

- **`local lines=$(wc -l < "$filename")`:** This line uses the `wc -l` command to count the number of lines in the file and stores the result in the variable `lines`. –

**`echo "The file '$filename'** has $lines lines."`: This line prints the number of lines in the file.
- `else`: This line starts the block of code that runs if the file does not exist.

- `**echo** "**The file '$filename' does not exist**."`: This line prints a message indicating that the file does not exist. –

The function `**c_line**` is called with three different filenames: **`file1.txt`, `file2.txt`,** You can replace these filenames with the names of the files you want to check.

4. **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").**

   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, follow these steps:

   1. Open a terminal.
   2. Create a new shell script file. You can name it `**create_files.sh**`.
   3. Open the file with a text editor and add the following code:

   #!/bin/bash

prajwalhonashetti143@gmail.com

mkdir -p TestDir

cd TestDir

for i in (1..10)
do

Filename=”File${i}.txt”
Echo “$Filename” > “$Filename”
done
echo “created 10 files with their filenames as content”

4. Save the file and close the editor.
5. Make the script executable by running the following command in the terminal:
``` **chmod +x create_files.sh** ```
6. Run the script by typing: ``` **./create_files.sh**

Explanation:

- The script sets the directory name TestDir using dir_name="TestDir".
- It creates the directory TestDir if it doesn't already exist using mkdir -p "$dir_name".
- Changes into the TestDir directory using cd "$dir_name".
- Loops from 1 to 10 (for I in (1…10)) to create files named File1.txt through File10.txt.
- For each file, it echoes its filename into the file itself (echo "$filename" > "$filename").
- After creating all the files, it moves back to the original directory using cd ...

5. **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.**
#!/bin/bash

To modify the script to handle errors and add a debugging mode, follow these steps:

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

 # 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

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

Explanation:

- The script checks if the first argument is **--debug (if ["$1" = "--debug"]).**
- If **--debug** is provided, it calls **enable_debug** function which sets **debug_mode=true** and prints a debug message.
- The **create_files** function is responsible for creating the directory and files.
- It first checks if the directory **TestDir** already exists (**if [-d "$dir_name"]).**
- If the directory already exists, it prints an error message and returns with a non-zero exit code (**return 1).**
- After creating the directory using **mkdir -p "$dir_name",** it checks if the directory creation was successful (**if [$? -ne 0]).**
- Inside the loop to create files (**for ((i = 1; i <= 10; i++))),** it creates each file and checks if the creation was successful (**if [$? -ne 0]).**
- If any error occurs during the directory creation or file creation, it prints an error message and returns with a non-zero exit code **(return 1).**
- The main script calls **create_files** function and checks its exit status using $?.
- If **create_files** exits with a zero status **($? -eq 0),** it prints "Files created successfully.".
- If **create_files** exits with a non-zero status, it prints "Failed to create files.".
```

prajwalhonashetti143@gmail.com

6. **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**

   To process a sample log file and extract lines containing "ERROR", then use `awk` to print the date, time, and error message, and finally perform some data processing with `sed`, follow these steps:
   1. Open the terminal.
   2. Create a sample log file named **`sample.log`** with some log entries. For example:

   2024-05-19 08:30:01 INFO: Starting application
   2024-05-19 08:30:10 ERROR: Database connection failed
   2024-05-19 08:31:15 ERROR: Out of memory
   2024-05-19 08:32:00 DEBUG: Received request from 192.168.1.100
   2024-05-19 08:32:05 ERROR: Disk full

   3. Create a new shell script file named `process_logs.sh`.
   4. 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/^[^:]*:[^:]*: //'
```

5. Save the file and close the editor.
6. Make the script executable by running the following command in the terminal:
```` **chmod +x process_logs.sh** ````

7. Run the script by typing: ``` **./process_logs.sh**```

Explanation:

- **if [! -f "$logfile"]: Checks if the log file exists.**
- **if [! -r "$logfile"]: Checks if the log file is readable.**
- **grep "ERROR" "$logfile": Searches for lines containing "ERROR" in the log file.**
- **awk -F ': ' '{print $1, $2}': Processes each line using awk:**
- **-F ': ': Sets the field separator to ": ".**
- **{print $1, $2}: Prints the date and time (first two fields).**
- **sed 's/^[^:]*:[^:]*: //': Uses sed to remove the date, time, and initial colon from each line containing "ERROR".**

7. **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.**

   To create a script that takes a text file and replaces all occurrences of "old_text" with "new_text" using `**sed`,** and outputs the result to a new file,

   follow these steps:
   1. Open a terminal.
   2. Create a new shell script file named `replace_text.sh`.
   3. 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
```
```

prajwalhonashetti143@gmail.com

```
    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."
```

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

Explanation: -
`if [ "$#" -ne 3 ]; then ... fi`: This checks if the correct number of arguments (3) are provided. If not, it prints the usage message and exits.
- **`INPUT_FILE="$1"`:** This assigns the first argument to the `INPUT_FILE` variable. - **`OLD_TEXT="$2"`:** This assigns the second argument to the `OLD_TEXT` variable. - **`NEW_TEXT="$3"`:** This assigns the third argument to the `NEW_TEXT` variable. - **`OUTPUT_FILE="output_$(basename "$INPUT_FILE**")"`: This constructs the output filename by prefixing `output_` to the basename of the input file. –
**`if [ ! -f "$INPUT_FILE" ];** then ... fi`: This checks if the input file exists. If not, it prints an error message and exits. –
`sed "s/${OLD_TEXT}/${NEW_TEXT}/g" "$INPUT_FILE" > "$OUTPUT_FILE"`: This uses `sed` to replace all occurrences of `OLD_TEXT` with `NEW_TEXT` in the input file and redirects the output to the new file. –
**`echo "Replaced all occurrences of '$OLD_TEXT' with '$NEW_TEXT' in '$INPUT_FILE'** and saved to '$OUTPUT_FILE'.": This informs the user about the completion of the operation.
With this script, you can easily replace text in a file and save the result to a new file.