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".

1. Create the Script:

- The script checks if a file named `myfile.txt` exists.
- It prints "File exists" if the file is found.
- It prints "File not found" if the file is not found.

Script content:

```sh
#!/bin/bash

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

2. Save the Script:

- The script is saved with the name `myfile.sh`.

3. Make the Script Executable:

```sh
chmod u+x myfile.sh
```

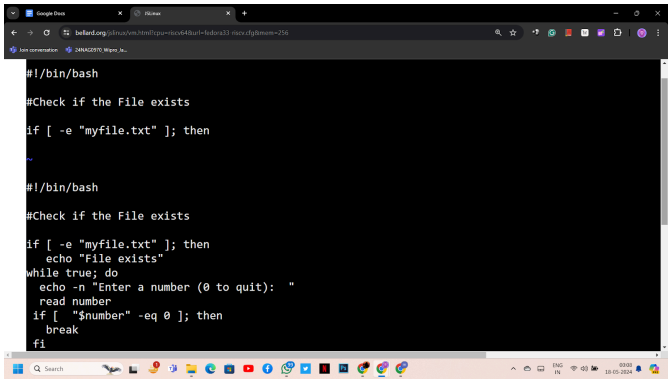
- This command makes the script file (`myfile.sh`) executable.

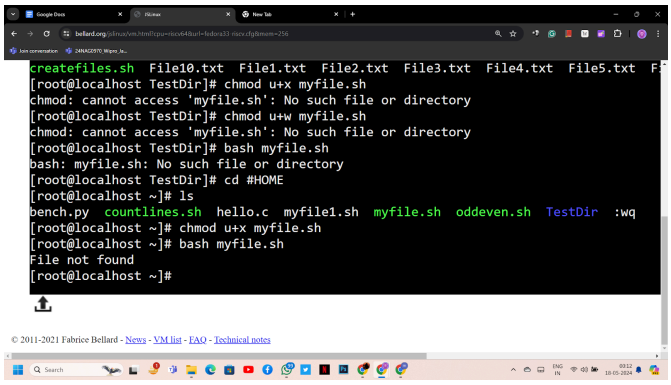
4. Run the Script:

```sh
bash myfile.sh
```

- This command runs the script.
- Output: Since `myfile.txt` does not exist in the directory, it prints "File not found".

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.

1. Script Creation:
- A script named `oddeven.sh` is created to determine if entered numbers are odd or even.

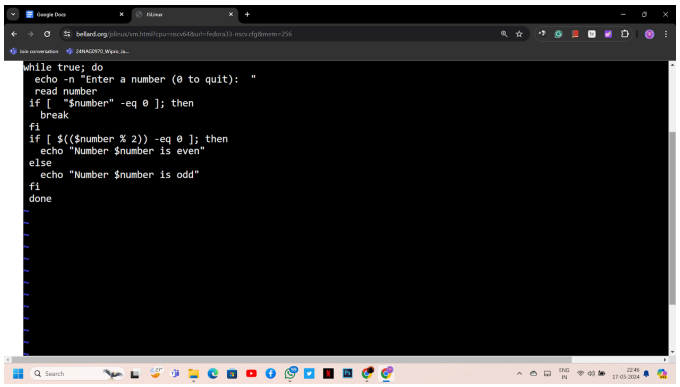
2. Making the Script Executable:
- The command `chmod +x oddeven.sh` is used to make the script executable.

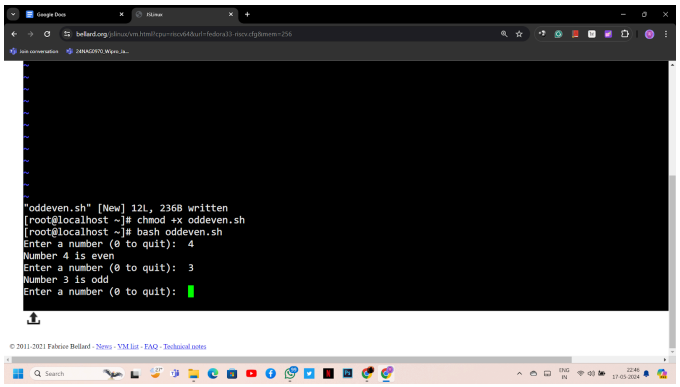
3. Running the Script:
- The command `bash oddeven.sh` is used to execute the script.

4. Script Execution:
- The script prompts the user to enter a number with the message "Enter a number (0 to quit):".
- If the user enters `4`, the script prints "Number 4 is even".
- If the user enters `3`, the script prints "Number 3 is odd".
- The process repeats until the user enters `0`, which terminates the script.

Summary:
- **Purpose:** The script checks if a number is odd or even until the user inputs `0`.
- Steps:
  1. Create and edit the script.
  2. Make the script executable.
  3. Run the script.
  4. Enter numbers as prompted.
  5. Script prints whether each number is odd or even.
  6. Enter `0` to exit the script.

Output 😀





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.

1. Define Function `count_lines`
- The function `count_lines` is defined to count the number of lines in a given file.
- It accepts one argument, which is the filename.

2. Declare Local Variable `filename`
- Within the function, the local variable `filename` is assigned the value of the first argument passed to the function.

3. Check if File Exists
- The function checks if the file specified by `filename` exists using a conditional statement.

4. If File Exists
- If the file exists, the local variable `line_count` is declared.
- The `wc -l` command is used to count the number of lines in the file, and the result is stored in `line_count`.
- A message is printed to the terminal indicating the number of lines in the file.

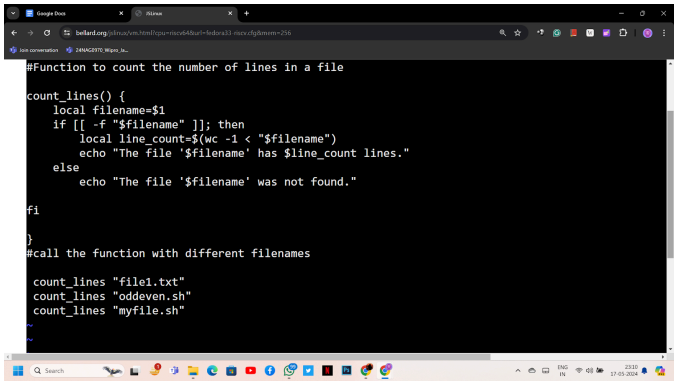
5. If File Does Not Exist
- If the file does not exist, a message is printed to the terminal indicating that the file was not found.

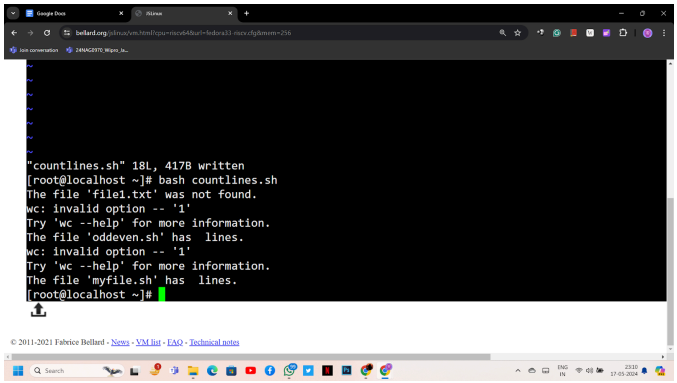
6. End of Conditional Statement
- The conditional statement is closed.

7. End of Function Definition
- The function definition is closed.

8. Call the Function with Different Filenames
- The function `count_lines` is called three times with different filenames: `file1.txt`, `oddeven.sh`, and `myfile.sh`.

Output 😀





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").

Commands and Outputs:

1. ls TestDir:
- Output: Error message "ls: cannot access 'TestDir': No such file or directory."
- Description: Attempted to list contents of a non-existent directory (TestDir).

2. cat Testdir/File.txt:
- Output: Error message "cat: Testdir/File.txt: No such file or directory."
- Description: Tried to display a file in a directory that doesn't exist (Testdir).

3. rmdir TestDir:
- Output: Error message "rmdir: failed to remove 'TestDir': No such file or directory."
- Description: Attempted to remove a non-existent directory (TestDir).

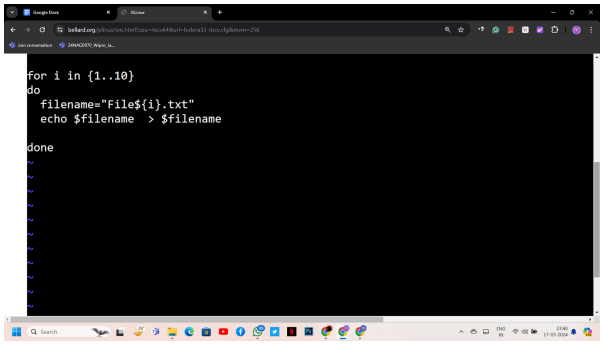
4. ls:
- Output: Lists files in the current directory: createfiles.sh, File10.txt, File1.txt, ..., File9.txt.
- Description: Showed existing files in the current directory.

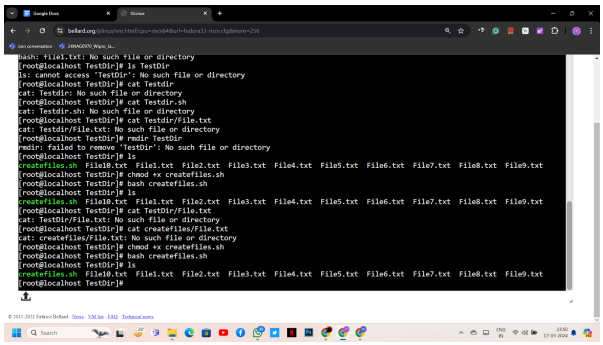
5. chmod +x createfiles.sh:
- Description: Made the createfiles.sh script executable.

6. ./createfiles.sh:
- Description: Executed the createfiles.sh script to create files File1.txt to File10.txt.

7. ls:
- Output: Lists files in the current directory again: createfiles.sh, File10.txt, File1.txt, ..., File9.txt.
- Description: Confirmed the presence of files created by the script.

Output 👍





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.

1. Debugging Mode:
Run the script with `-d` or `--debug` to enable debugging mode. Example: `./script.sh -d directory_name`

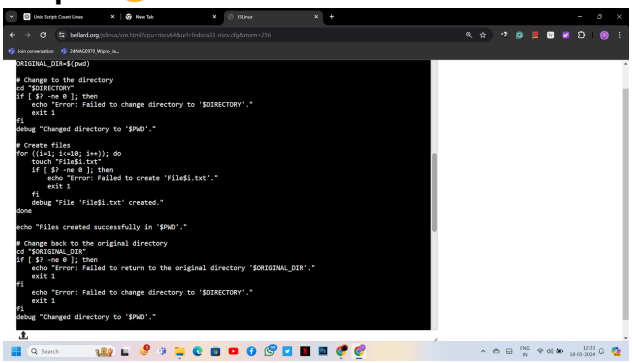
2. Directory Handling:
1.Check if a directory name is provided. Example: `if [ -z "$1" ]; then echo "Error: Directory name not provided."; exit 1; fi`
2.Check if the directory exists or create it: `if [ -d "$directory_name" ]; then echo "Directory already exists."; else mkdir "$directory_name" || { echo "Error: Failed to create directory."; exit 1; }; fi`

3. File Creation:
1.Loop to create ten files: `for i in {1..10}; do touch "$directory_name/File$i.txt"; done`
2.Handle existing files: `for i in {1..10}; do [ -f "$directory_name/File$i.txt" ] && echo "File File$i.txt already exists." || touch "$directory_name/File$i.txt"; done`

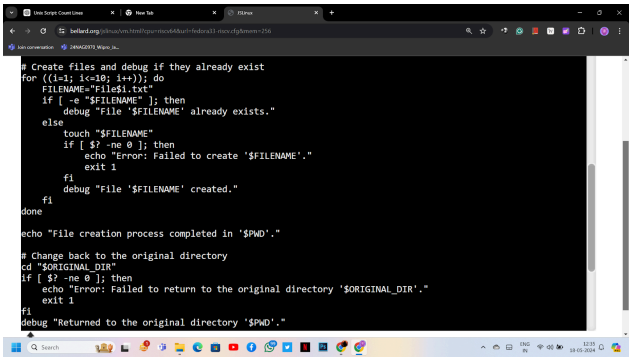
4. Debugging Messages:
Use `echo` statements to print debug messages throughout the script. Example: `echo "Debug: Directory changed to $directory_name"`

5. Completion and Cleanup:
1. Print completion message: `echo "Files created successfully in $directory_name"`
2. Return to original directory: `cd "$original_dir" || { echo "Error: Failed to change back to original directory."; exit 1; }`

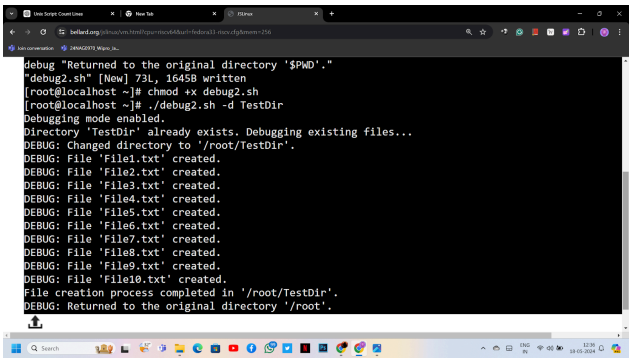
Output 😳



```
ORIGINAL_DIR=$(pwd)

# Change to the directory
cd "$DIRECTORY"
if [ $? -ne 0 ]; then
    echo "Error: Failed to change directory to '$DIRECTORY'."
    exit 1
fi
debug "Changed directory to '$PWD'."

# Create files
for ((i=1; i<=10; i++)); do
    touch "File$i.txt"
    if [ $? -ne 0 ]; then
        echo "Error: Failed to create 'File$i.txt'."
        exit 1
    fi
    debug "File 'File$i.txt' created."
done

echo "Files created successfully in '$PWD'."

# Change back to the original directory
cd "$ORIGINAL_DIR"
if [ $? -ne 0 ]; then
    echo "Error: Failed to return to the original directory '$ORIGINAL_DIR'."
    exit 1
fi

    echo "Error: Failed to change directory to '$DIRECTORY'."
    exit 1
fi
debug "Changed directory to '$PWD'."
```



```
# Create files and debug if they already exist
for ((i=1; i<=10; i++)); do
    FILENAME="File$i.txt"
    if [ -e "$FILENAME" ]; then
        debug "File '$FILENAME' already exists."
    else
        touch "$FILENAME"
        if [ $? -ne 0 ]; then
            echo "Error: Failed to create '$FILENAME'."
            exit 1
        fi
        debug "File '$FILENAME' created."
    fi
done

echo "File creation process completed in '$PWD'."

# Change back to the original directory
cd "$ORIGINAL_DIR"
if [ $? -ne 0 ]; then
    echo "Error: Failed to return to the original directory '$ORIGINAL_DIR'."
    exit 1
fi
debug "Returned to the original directory '$PWD'."
```



```
debug "Returned to the original directory '$PWD'."
"debug2.sh" [New] 73L, 1645B written
[root@localhost ~]# chmod +x debug2.sh
[root@localhost ~]# ./debug2.sh -d TestDir
Debugging mode enabled.
Directory 'TestDir' already exists. Debugging existing files...
DEBUG: Changed directory to '/root/TestDir'.
DEBUG: File 'File1.txt' created.
DEBUG: File 'File2.txt' created.
DEBUG: File 'File3.txt' created.
DEBUG: File 'File4.txt' created.
DEBUG: File 'File5.txt' created.
DEBUG: File 'File6.txt' created.
DEBUG: File 'File7.txt' created.
DEBUG: File 'File8.txt' created.
DEBUG: File 'File9.txt' created.
DEBUG: File 'File10.txt' created.
File creation process completed in '/root/TestDir'.
DEBUG: Returned to the original directory '/root'.
```

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

1. Script Creation:
- A script named `error3.sh` is created, consisting of 19 lines and 628 bytes.

2. Making the Script Executable:
- The command `chmod +x error3.sh` is used to make the script executable.

3. Running the Script:
- The script is executed using the command `./error3.sh`.

4. Script Output:
- Upon execution, the script outputs specific log entries indicating errors. The output shown includes:
  - `2024-05-20 08:30:00 full`
  - `2024-05-22 12:00:45 connection failed`
  - `2024-05-23 15:30:00 timeout`

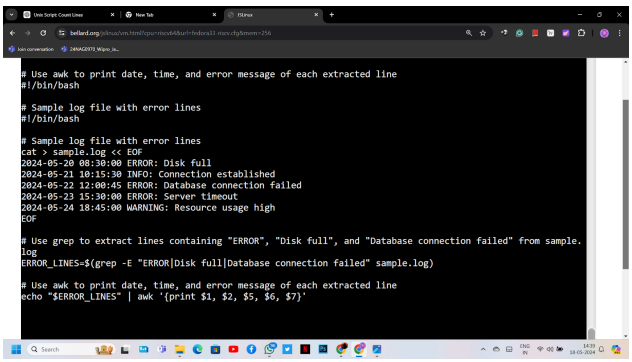
Detailed Breakdown of `error3.sh`:

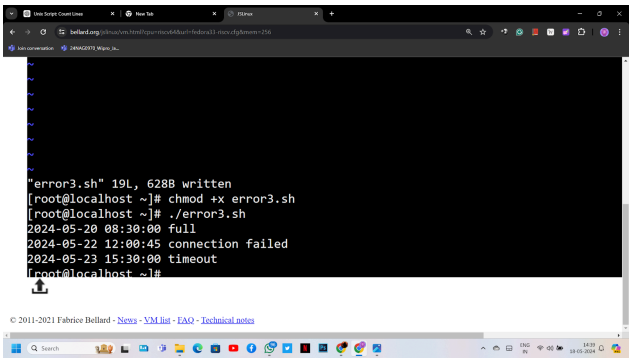
1. Creating a Log File:
- The script likely includes a section that creates or references a `sample.log` file containing various log messages.

2. Finding Errors:
- The script scans the `sample.log` file for Examples of such error keywords could be "full" (potentially indicating a disk full error), "connection failed," and "timeout."

3. Execution and Output:
- When the script is executed:
  - It reads through the `sample.log` file.
  - It identifies lines containing the specified error keywords ("full," "connection failed," and "timeout").
  - It outputs these lines with their respective timestamps and error msgs

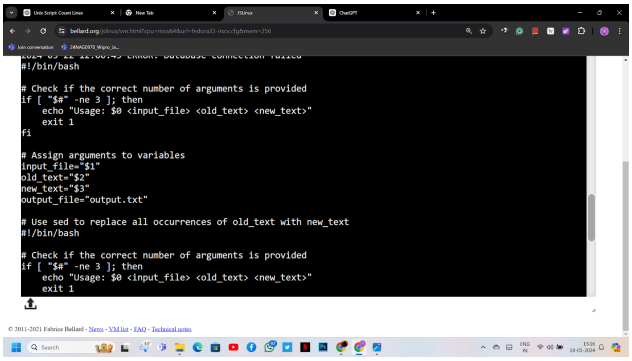
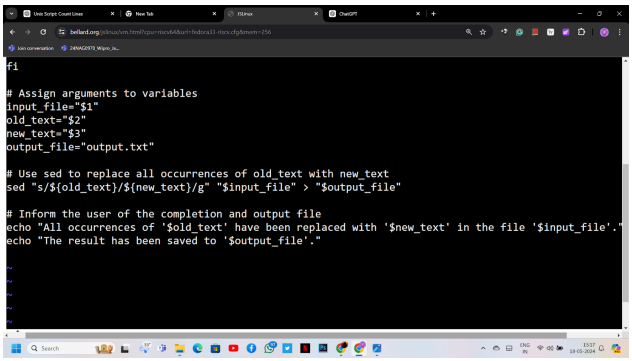
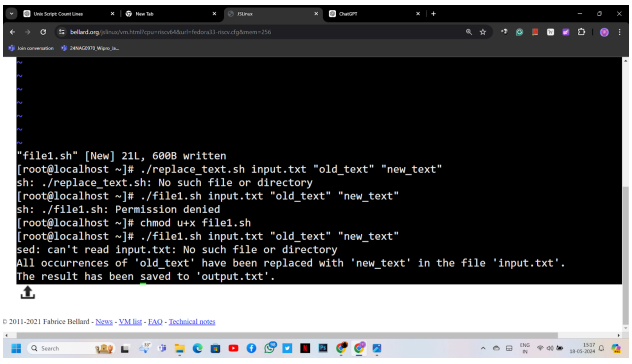
Output 😀





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.

Description	Output
1. Create the Script: - Create a file named `file1.sh` and write your script in it. **2. Make the Script Executable:** - Run the command `chmod +xfile1.sh` to make the script executable. **3. Run the Script:** - Execute the script with three arguments: the input file, the text to replace, and the new text, using the command `./file1.sh.sh input.txt old_string new_string`. **4. Script Actions:** - The script checks if three arguments are provided. If not, it shows a usage message and exits. - It assigns the input arguments to variables. - It uses `sed` to replace all occurrences of the old text with the new text in the input file and saves the result to `output.txt`. - It informs you that the replacement is done and the result is saved in `output.txt`.	  

Manishaanaparthi123@gmail.com