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".

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

Save this script in a file,

for example, `check_file.sh`, and make it executable using the command

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

You can run this script in the terminal by navigating to the directory containing the script and then executing it:

```
./check_file.sh
```

If `myfile.txt` exists in the same directory as the script, it will print "File exists"; otherwise, it will print "File not found".

Code:

```
rps@rps: ~/Desktop/BashScript
  GNU nano 4.8                    assignment1.sh
#!/bin/bash

filename="file1.txt"

if [ -e "$filename"]; then
        echo "File exist"
else
        echo "File not Found"
fi

                              [ Wrote 9 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Output:

```
rps@rps: ~/Desktop/BashScript
rps@rps:~/Desktop/BashScript$ ./assignment1.sh
File not Found
rps@rps:~/Desktop/BashScript$ ls
Arthimetic.sh        assignment3.sh  hello_world.sh  unsetvariable.sh
assignment1.sh       demo1.sh        ifdemo1.sh      userinput.sh
assignment1.sh.save  function.sh     nano.save       variable.sh
rps@rps:~/Desktop/BashScript$ ./assignment1.sh
File exist
rps@rps:~/Desktop/BashScript$
```

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.

```
#!/bin/bash
while true; do
  read -p "Enter a number (0 to exit): " num
  if [ "$num" -eq 0 ]; then
    break
  elif [ "$(($num % 2))" -eq 0 ]; then
    echo "$num is even."
  else
    echo "$num is odd."
  fi
done
```

Code:

```
rps@rps: ~/Desktop/BashScript
  GNU nano 4.8                    assignment2.sh
#!/bin/bash

while true; do
        read -p "Enter  number ( 0 to exits ): " num
        if [ "$num" -eq  0 ]; then
                break;
        elif [ "$(($num % 2))" -eq 0 ]; then
                echo "$num is even."
        else
                echo "$num is odd."
        fi
done

                        [ Wrote 12 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Output:

```
rps@rps: ~/Desktop/BashScript
rps@rps:~/Desktop/BashScript$ chmod 777 assignment2.sh
rps@rps:~/Desktop/BashScript$ ./assignment2.sh
Enter  number ( 0 to exits ): 2
2 is even.
Enter  number ( 0 to exits ): 4
4 is even.
Enter  number ( 0 to exits ): 3
3 is odd.
Enter  number ( 0 to exits ): 2
2 is even.
Enter  number ( 0 to exits ): 9
9 is odd.
Enter  number ( 0 to exits ): 7
7 is odd.
Enter  number ( 0 to exits ): 0
rps@rps:~/Desktop/BashScript$
```

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.

```
#!/bin/bash
count_lines() {
    filename="$1"
    if [ -f "$filename" ]; then
        lines=$(wc -l < "$filename")
        echo "Number of lines in $filename: $lines"
    else
        echo "File $filename not found."
    fi
}
# Call the function with different filenames
count_lines "file1.txt"
count_lines "file2.txt"
```

Code:

```
rps@rps: ~/Desktop/BashScript
  GNU nano 4.8                    assignment3.sh
#!/bin/bash

# Define a function to count lines in a file
count_lines() {
    file=$1
    num_lines=$(wc -l < "$file")
    echo "Number of lines in $file: $num_lines"

    echo "Data in $file:"
    while read -r line; do
        echo "$line"
    done < "$file"
}

# Call the function with different file names
count_lines "userinput.sh"
count_lines "ifdemo1.sh"
count_lines "variable.sh"

                          [ Read 19 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Output:

```
rps@rps: ~/Desktop/BashScript
rps@rps:~/Desktop/BashScript$ ./assignment3.sh
Number of lines in userinput.sh: 10
Number of lines in ifdemo1.sh: 15
Number of lines in variable.sh: 11
rps@rps:~/Desktop/BashScript$
```

Assignment 4:

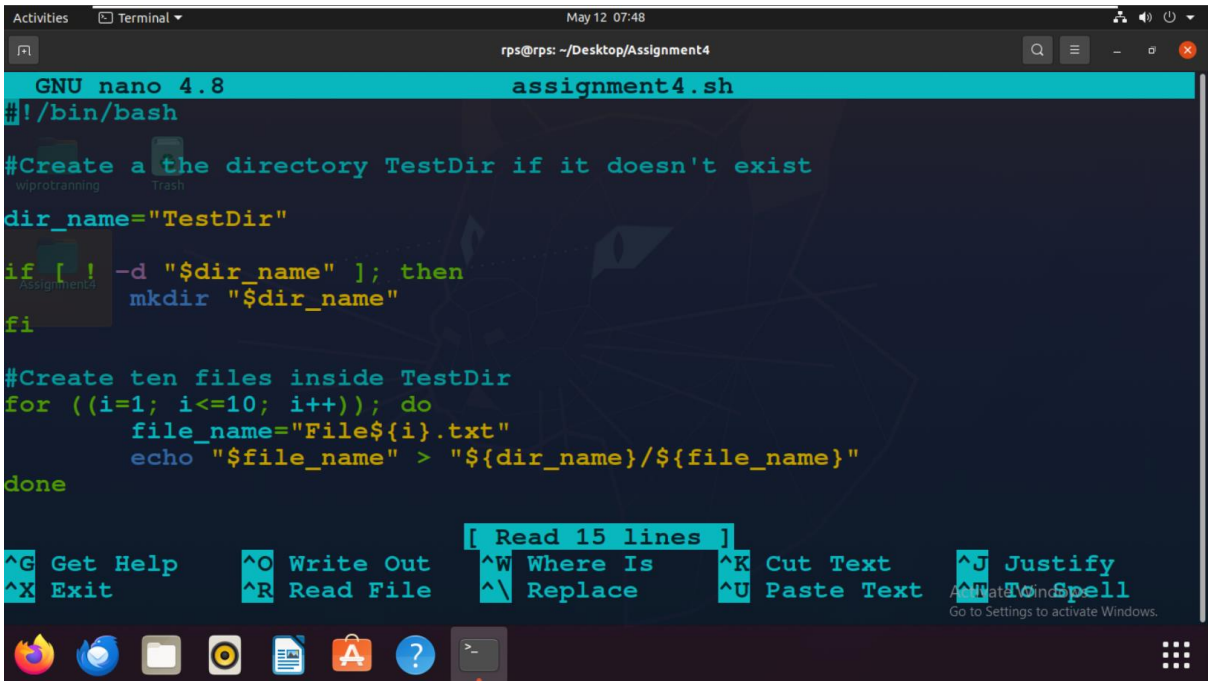
Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

```
#!/bin/bash
# Create the directory TestDir if it doesn't exist
dir_name="TestDir"
if [ ! -d "$dir_name" ]; then
  mkdir "$dir_name"
fi
# Create ten files inside TestDir
for (( i=1; i<=10; i++ )); do
  file_name="File${i}.txt"
  echo "$file_name" > "${dir_name}/${file_name}"
done
```

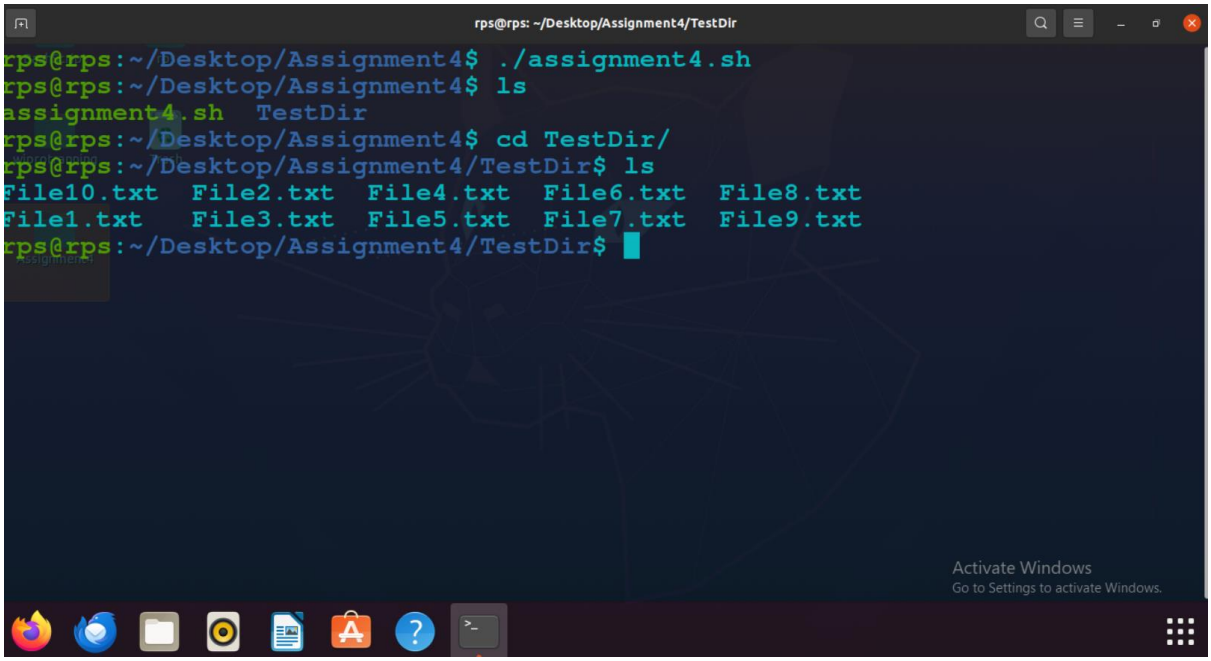
Save the above script to a file, let's say `create_files.sh`, then make it executable using `chmod +x create_files.sh`, and finally run it using `./create_files.sh`.

This script first checks if the directory `TestDir` exists, and if not, it creates it using `mkdir`. Then, it uses a loop to create ten files (`File1.txt` to `File10.txt`) inside `TestDir`, each with its filename as content using `echo`.

Code:


Activities
Terminal
May 12 07:48
rps@rps: ~/Desktop/Assignment4
GNU nano 4.8
assignment4.sh
#!/bin/bash
#Create a the directory TestDir if it doesn't exist
dir_name="TestDir"
if [! -d "$dir_name"]; then
mkdir "$dir_name"
fi
#Create ten files inside TestDir
for ((i=1; i<=10; i++)); do
file_name="File${i}.txt"
echo "$file_name" > "${dir_name}/${file_name}"
done
[Read 15 lines]
^G Get Help
^O Write Out
^W Where Is
^K Cut Text
^J Justify
^X Exit
^R Read File
^\ Replace
^U Paste Text
^T To Spell
Go to Settings to activate Windows.

Output:


rps@rps: ~/Desktop/Assignment4/TestDir
rps@rps:~/Desktop/Assignment4$ ./assignment4.sh
rps@rps:~/Desktop/Assignment4$ ls
assignment4.sh TestDir
rps@rps:~/Desktop/Assignment4$ cd TestDir/
rps@rps:~/Desktop/Assignment4/TestDir$ ls
File10.txt File2.txt File4.txt File6.txt File8.txt
File1.txt File3.txt File5.txt File7.txt File9.txt
rps@rps:~/Desktop/Assignment4/TestDir$
Activate Windows
Go to Settings to activate Windows.

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
# Debug mode flag
debug=false
# Function to print debug messages
debug_msg() {
    if [ "$debug" = true ]; then
        echo "DEBUG: $1"
    fi
}
# Create the directory TestDir if it doesn't exist
dir_name="TestDir"
if [ -d "$dir_name" ]; then
    echo "Error: Directory '$dir_name' already exists."
    exit 1
fi

if ! mkdir "$dir_name"; then
    echo "Error: Unable to create directory '$dir_name'. Check permissions."
    exit 1
fi
```

```
debug_msg "Created directory '$dir_name'."

# Create ten files inside TestDir
for (( i=1; i<=10; i++ )); do
    file_name="File${i}.txt"
    if ! echo "$file_name" > "${dir_name}/${file_name}"; then
        echo "Error: Unable to create file '$file_name' inside '$dir_name'. Check permissions."
        exit 1
    fi
    debug_msg "Created file '$file_name' inside '$dir_name'."
done

echo "Files created successfully in '$dir_name'."
```

This script checks for errors such as the directory already existing or lacking permissions to create files. It also includes a debugging mode that prints additional information when enabled.

```
  GNU nano 4.8                                   assignment5.sh
#!/bin/bash

#Debug mode Flag
debug=true

#Function to print debug message
debug_msg() {
if [ "$debug" = true ]; then
        echo "DEBUG: $1"
fi

}

# Create the directory TestDir if it doesn't exist
dir_name="TestDir"

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


if [ ! mkdir "$dir_name" ]; then

        echo "Error: Unable to create a directory '$dir_name'. Check Permissions."
        exit 1
fi

debug_msg "Created directory '$dir_name'."

#Create  ten  files inside TestDir
for  (( i=1; i<=10; i++ )); do
        file_name="File${i}.txt"
        if [ ! echo "$file_name" > "${dir_name}/${file_name}"; then

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

```
  GNU nano 4.8                                   assignment5.sh
fi

}

# Create the directory TestDir if it doesn't exist
dir_name="TestDir"

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


if [ ! mkdir "$dir_name" ]; then

        echo "Error: Unable to create a directory '$dir_name'. Check Permissions."
        exit 1
fi

debug_msg "Created directory '$dir_name'."

#Create  ten  files inside TestDir
for  (( i=1; i<=10; i++ )); do
        file_name="File${i}.txt"
        if [ ! echo "$file_name" > "${dir_name}/${file_name}"; then
        echo "Error: Unable to create file '$file_name' inside '$dir_name'. Check Permissions."
        exit 1
        fi

        debug_msg "Created file '$file_name' inside '$dir_name'."
done

echo "File Created Successfully in '$dir_name'."

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

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.

```
#!/bin/bash

# Define the log file path

log_file="sample.log"

# Use grep to extract lines containing "ERROR" and then use awk to print date, time, and error message

grep "ERROR" "$log_file" | awk '{print $1, $2, $NF}'
```

Save the above script to a file, let's say `extract_errors.sh`, then make it executable using `chmod +x extract_errors.sh`, and finally run it using `./extract_errors.sh`.

In this script:

- `grep "ERROR" "$log_file"` filters the log file and extracts only the lines that contain the string "ERROR".

- `awk '{print $1, $2, $NF}'` takes each line outputted by `grep` and uses `awk` to print the first field (date), second field (time), and the last field (error message) of each line.

Code:

```
  GNU nano 4.8                    extract_error.sh
#!/bin/bash

#Define the  log  file Path
log_file="sample.log"

#use grep to extract lines cotainings "Error" and then use awk to print date time and error  message

grep  "Error"  "$log_file" | awk  '{print $1, $2. $NF}'

                                   [ Read 9 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

Output:

```
rps@rps:~/Desktop/Assignment4$ nano extract_error.sh
rps@rps:~/Desktop/Assignment4$ ./extract_error.sh
2024-5-12 8:00:00wrong
2024-5-12 8:50:00occured
2024-5-12 8:50:00detected
rps@rps:~/Desktop/Assignment4$
```

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.

```
#!/bin/bash
# Check if the correct number of arguments are provided
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 input_file old_text new_text"
    exit 1
fi
input_file=$1
old_text=$2
new_text=$3
output_file="${input_file}_modified.txt"
# Use sed to perform the replacement and write the output to a new file
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"
echo "Replacement complete. Modified text saved in $output_file"
```

Save this script in a file, let's say `replace_text.sh`, and then you can run it with three arguments: the input file, the old text you want to replace, and the new text you want to replace it with. For example:

```
./replace_text.sh input.txt old_text new_text
```

Replace `input.txt` with the actual name of your input file and `old_text` and `new_text` with the text you want to replace and the text you want to replace it with.