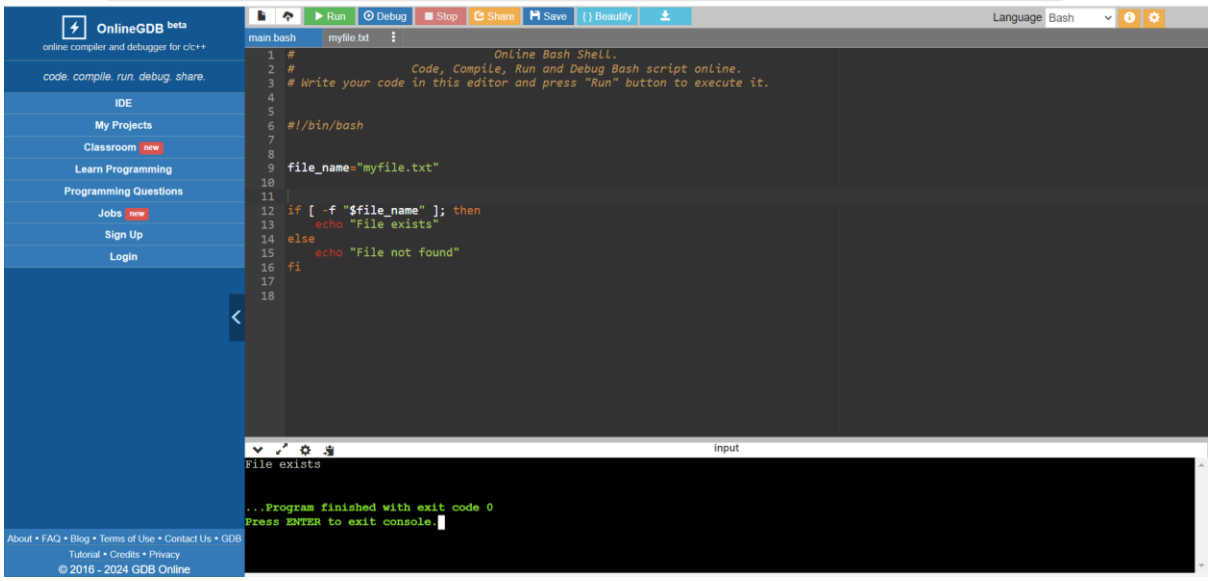
Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Code:

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

output:



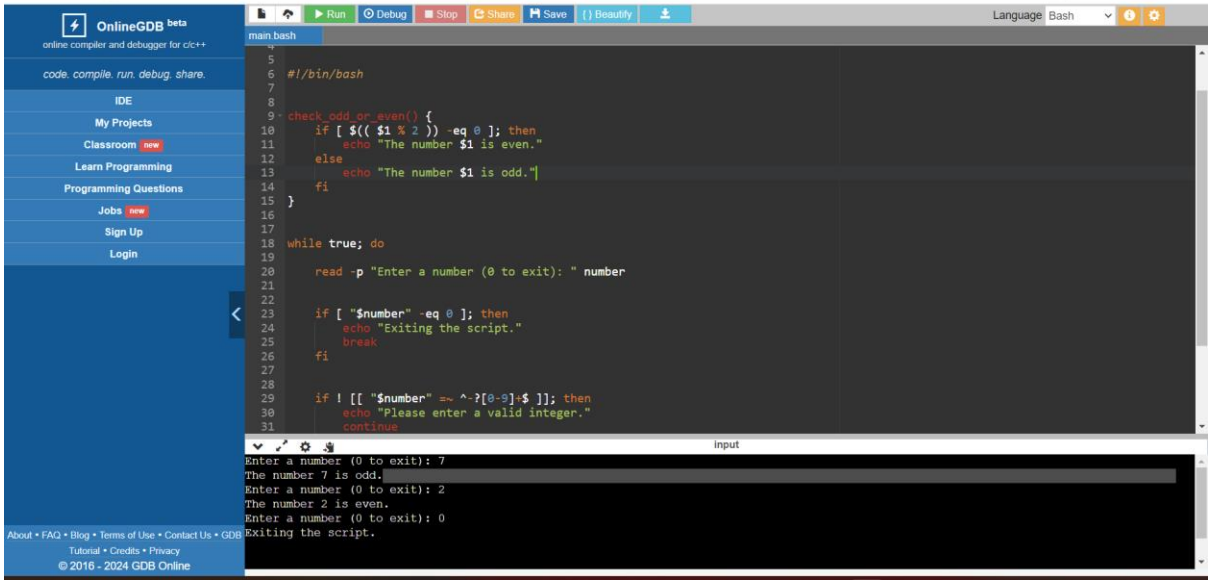
Assignment 2: Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.

Code:

```
#!/bin/bash
check_odd_or_even() {
   if [ $(( $1 % 2 )) -eq 0 ]; then
      echo "The number $1 is even."
   else
      echo "The number $1 is odd."
   fi
}
while true; do
  read -p "Enter a number (0 to exit): " number
```

```
 if [ "$number" -eq 0 ]; then
      echo "Exiting the script."
      break
   fi
 if ! [[ "$number" =~ ^-?[0-9]+$ ]]; then
      echo "Please enter a valid integer."
      continue
   fi
check_odd_or_even "$number"
done
```

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.

Code:

Step1:

nano count_line.sh

this command is use for the creating script file

after writing the script

use CTRL+O to save the file

press ENTER

For exit from the file PRESS

CTRL+X

The Script is:

```
#!/bin/bash
count_lines() {
```

```
 local filename=$1
 if [ -f "$filename" ]; then
 local line_count=$(wc -l < "$filename")
 echo "The file '$filename' has $line_count lines."
 else
 echo "Error: The file '$filename' does not exist."
 fi
}
count_lines "file1.txt"
count_lines "file2.txt"
count_lines "file3.txt"
```

Step3 :

Use command

chmod +x count_line.sh

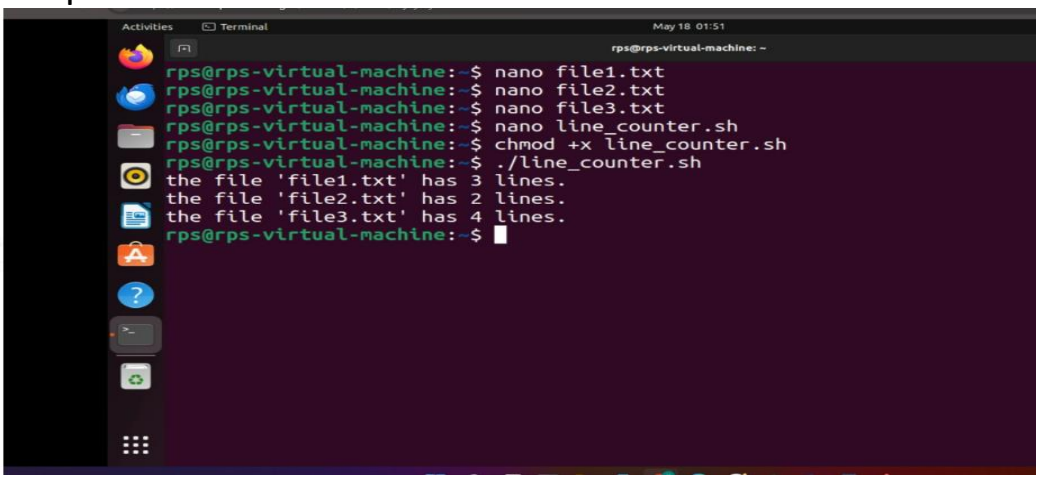
This command is essential because it sets the necessary permissions on the file to allow it to be run as a program in your Linux environment

Step4:

Use ./count_line.sh

To excute the script

Output:

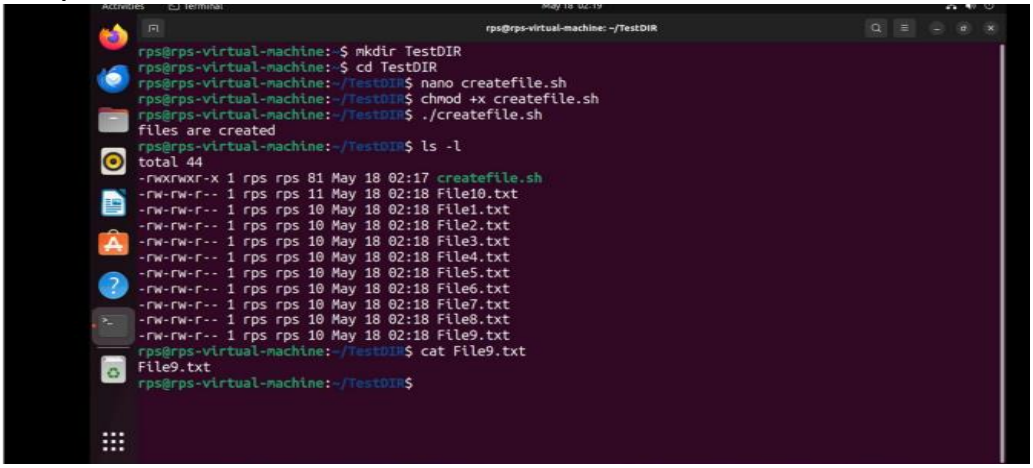


Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

Code: Use nano crtaefile10.sh
To create 10 file with same content as file name in the file
Script:

```
for i in {1..10}; do
 f="File$i.txt"
 echo $f > "$f"
done
echo " files created."
```

Output:



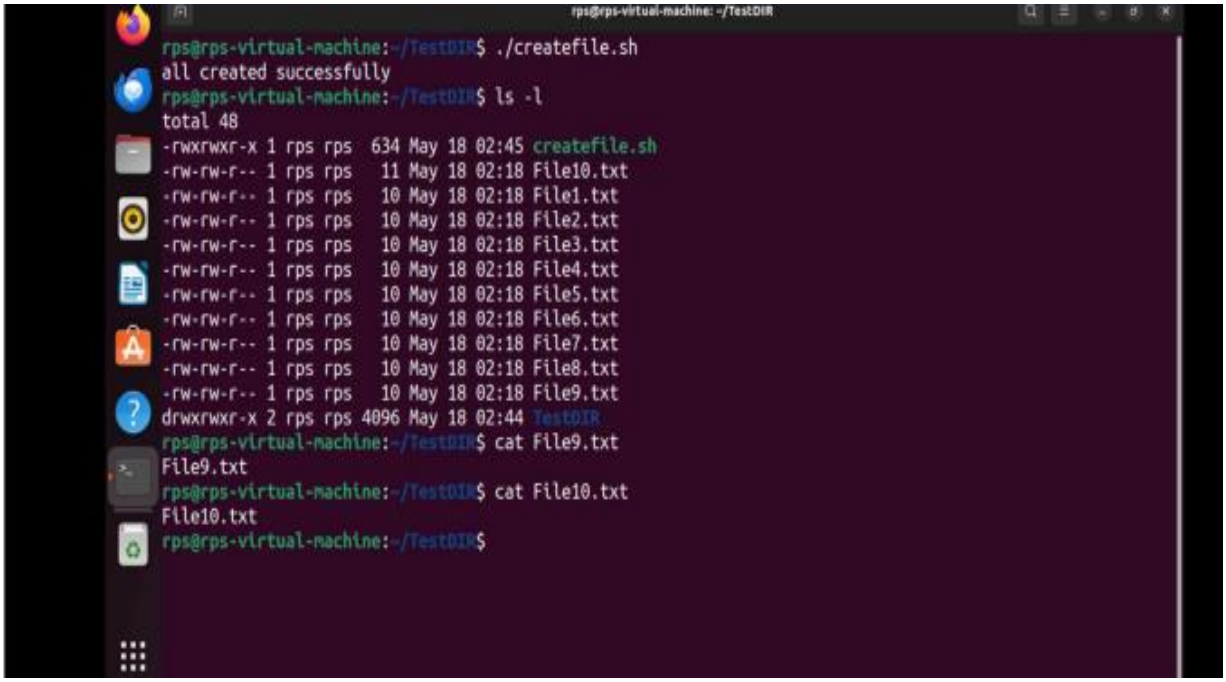
Assignment 5: Modify the script to handle errors, such as the directory already existing or lacking permissions to create files. Add a debugging mode that prints additional information when enabled.

Code:

```
#!/bin/bash
if [[ $1 == "debug" ]]; then
 debug_mode=true
else
 debug_mode=false
fi
debuglog() {
 if [ "$debug_mode" = true ]; then
 echo "DEBUG: $1"
 fi
}
dir ="TestDir"
```

```
if [ ! -d "$dir " ]; then
 mkdir "$dir "
 debuglog "Directory '$dir created successfully."
else
 debuglog "Directory '$dir already exists, skipping creation."
fi
if cd "$dir "; then
 debuglog "Changed directory to $dir."
else
 echo "Error: Failed to change directory to $dir . Check
permissions."
 exit 1
fi
for i in {1..10}; do
 f="File$i.txt"
 if echo "$f" > "$f"; then
 debuglog "Created and wrote to $f."
 else
 echo "Error: Failed to write to $f."
 exit 1
 fi
done
echo "all created successfull."
```

Output:



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

Code:

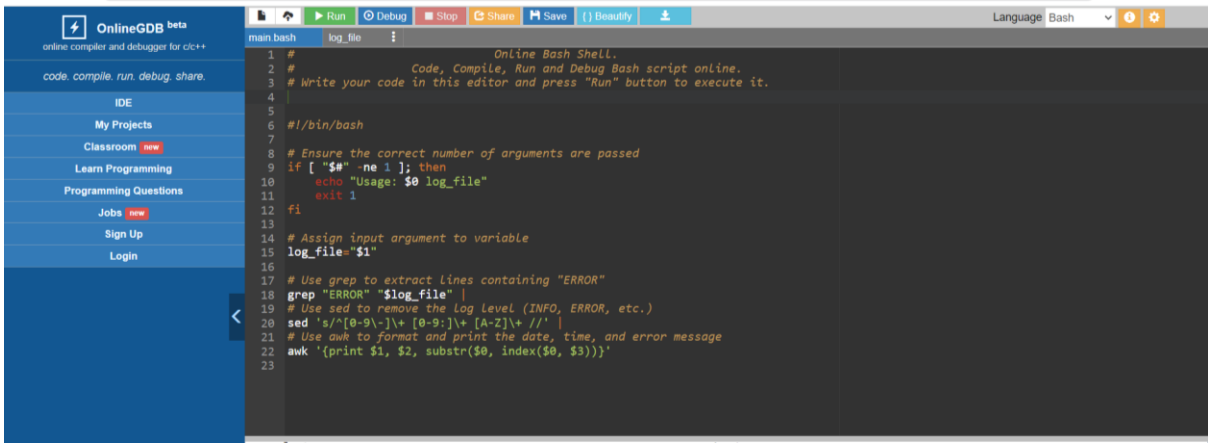
```
#!/bin/bash

# Ensure the correct number of arguments are passed
if [ "$#" -ne 1 ]; then
   echo "Usage: $0 log_file"
   exit 1
fi

# Assign input argument to variable
log_file="$1"

# Use grep to extract lines containing "ERROR"
grep "ERROR" "$log_file" |
# Use sed to remove the log level (INFO, ERROR, etc.)
sed 's/^[0-9\-]\+ [0-9:]\+ [A-Z]\+ //' |
# Use awk to format and print the date, time, and error message
awk '{print $1, $2, substr($0, index($0, $3))}'
```

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.

Code:

```
#!/bin/bash

if [ "$#" -ne 3 ]; then
  echo "Usage: $0 input_file old_text new_text"
  exit 1
fi

input_file=$1
old_text=$2
new_text=$3
output_file="output_${input_file}"

if [ ! -f "$input_file" ]; then
  echo "Error: Input file '$input_file' not found!"
  exit 1
fi

sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

# Output success message
echo "Replaced all occurrences of '$old_text' with '$new_text' in '$input_file' and saved to '$output_file'"
```

output:

onlinegdb.com/online_bash_shell

OnlineGDB beta
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login

Run | Debug | Stop | Share | Save | { } Beautify | Language Bash

main.bash | myfile.txt

```bash
# Write your code in this editor and press "Run" button to execute it.

#!/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

# Assign input arguments to variables
input_file=$1
old_text=$2
new_text=$3
output_file="output_${input_file}"

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

# Use sed to replace all occurrences of old_text with new_text and save the result to a new file
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

# Output success message
echo "Replaced all occurrences of '$old_text' with '$new_text' in '$input_file' and saved to '$output_file'"
```

Input