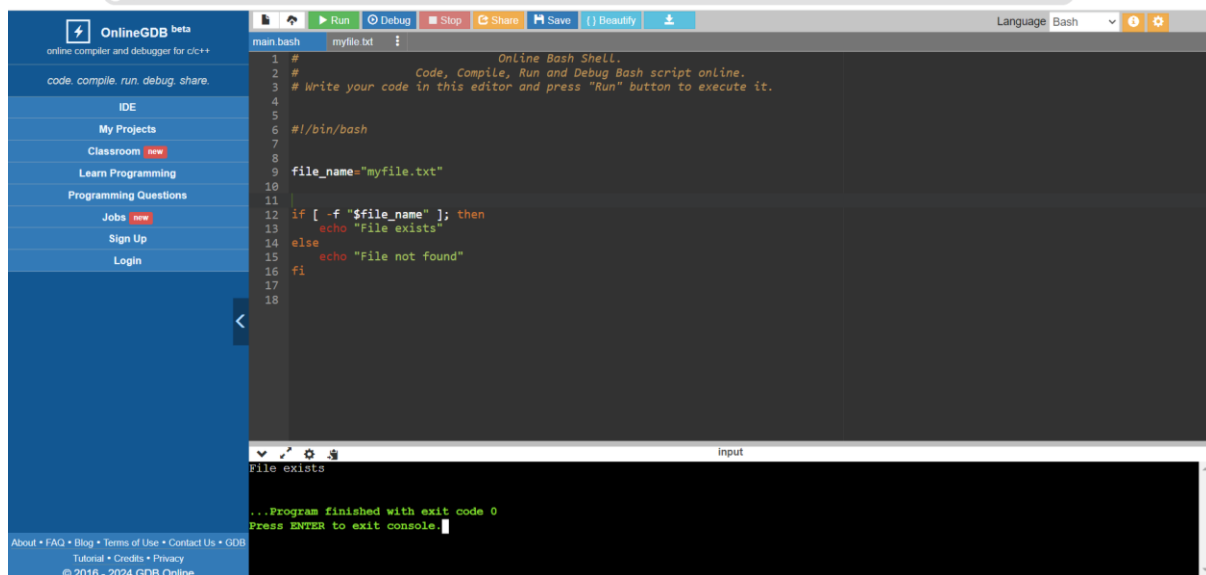


Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Code:

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

output:

The screenshot shows the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'code', 'compile', 'run', 'debug', 'share', 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Sign Up', and 'Login'. The main area is a dark-themed code editor with a file named 'myfile.txt' open. It contains a Bash script that checks for the existence of 'myfile.txt' and prints 'File exists'. Below the editor is a console window showing the output 'File exists' and a message '...Program finished with exit code 0'. The top of the interface has a toolbar with buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', and 'Beautify', along with a language dropdown set to 'Bash'.

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:

The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links like 'My Projects', 'Classroom', 'Learn Programming', etc. The main area displays a Bash script with line numbers 4 to 31. The script defines a function `check_odd_or_even` and a `while true; do` loop that prompts the user for a number. The output at the bottom shows the script being executed with inputs 7, 2, and 0, resulting in 'The number 7 is odd.', 'The number 2 is even.', and 'Exiting the script.' respectively.

```

4  #!/bin/bash
5
6
7
8
9  check_odd_or_even() {
10     if [ $((($1 % 2)) -eq 0 ]; then
11         echo "The number $1 is even."
12     else
13         echo "The number $1 is odd."
14     fi
15 }
16
17
18 while true; do
19     read -p "Enter a number (0 to exit): " number
20
21
22
23     if [ "$number" -eq 0 ]; then
24         echo "Exiting the script."
25         break
26     fi
27
28     if ! [[ "$number" =~ ^-[0-9]+$ ]]; then
29         echo "Please enter a valid integer."
30         continue
31

```

Enter a number (0 to exit): 7
The number 7 is odd.
Enter a number (0 to exit): 2
The number 2 is even.
Enter a number (0 to exit): 0
Exiting the script.

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
```

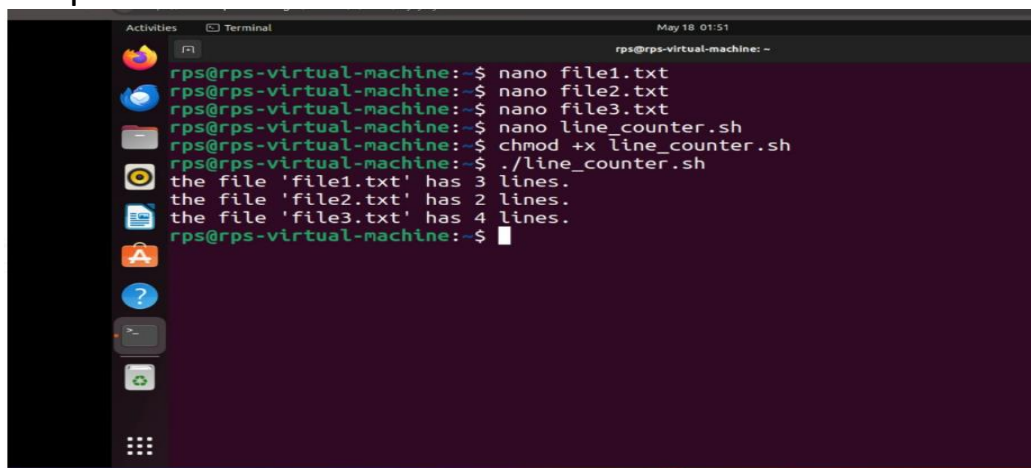
This command is essential because it sets the necessary permissions on the file to allow it to be run as a program in your Linux environment

Step4:

Use `./count_line.sh`

To excute the script

Output:



```

rps@rps-virtual-machine:~$ nano file1.txt
rps@rps-virtual-machine:~$ nano file2.txt
rps@rps-virtual-machine:~$ nano file3.txt
rps@rps-virtual-machine:~$ nano line_counter.sh
rps@rps-virtual-machine:~$ chmod +x line_counter.sh
rps@rps-virtual-machine:~$ ./line_counter.sh
the file 'file1.txt' has 3 lines.
the file 'file2.txt' has 2 lines.
the file 'file3.txt' has 4 lines.
rps@rps-virtual-machine:~$

```

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").

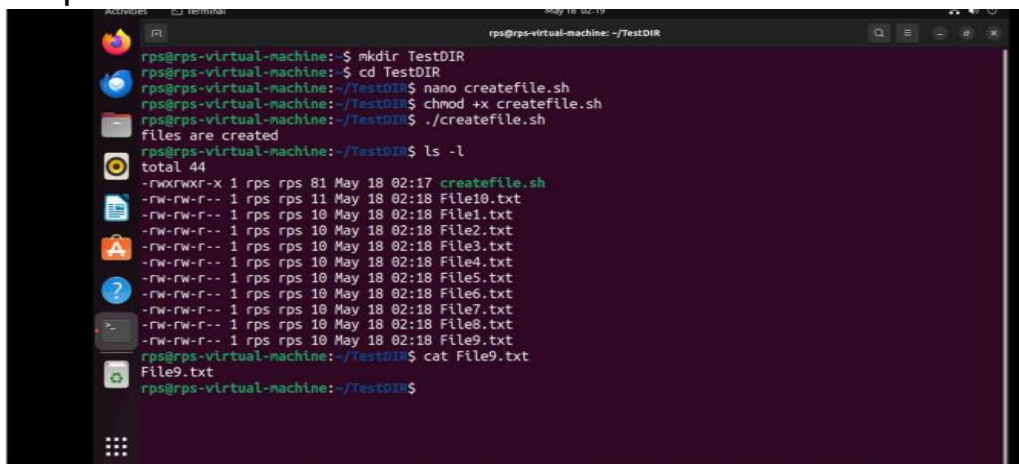
Code: Use nano crtaefile10.sh

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:

A screenshot of a terminal window on a Linux system. The user is in a directory named 'TestDIR'. They run 'mkdir TestDIR', 'cd TestDIR', and 'nano createfile.sh'. The script 'createfile.sh' is shown with the following content: 'for i in {1..10}; do', 'f="File\$i.txt"', 'echo \$f > "\$f"', 'done', 'echo " files created."'. The user runs 'chmod +x createfile.sh' and then './createfile.sh'. The output shows 'files are created' and 'rps@rps-virtual-machine:~/TestDIR\$ ls -l'. The output of 'ls -l' shows a total of 44 bytes, with 10 files listed: 'createfile.sh' (81 bytes) and 'File1.txt' through 'File10.txt' (11 bytes each). The user then runs 'cat File9.txt' and the output is 'File9.txt'. The prompt 'rps@rps-virtual-machine:~/TestDIR\$' is shown at the bottom.

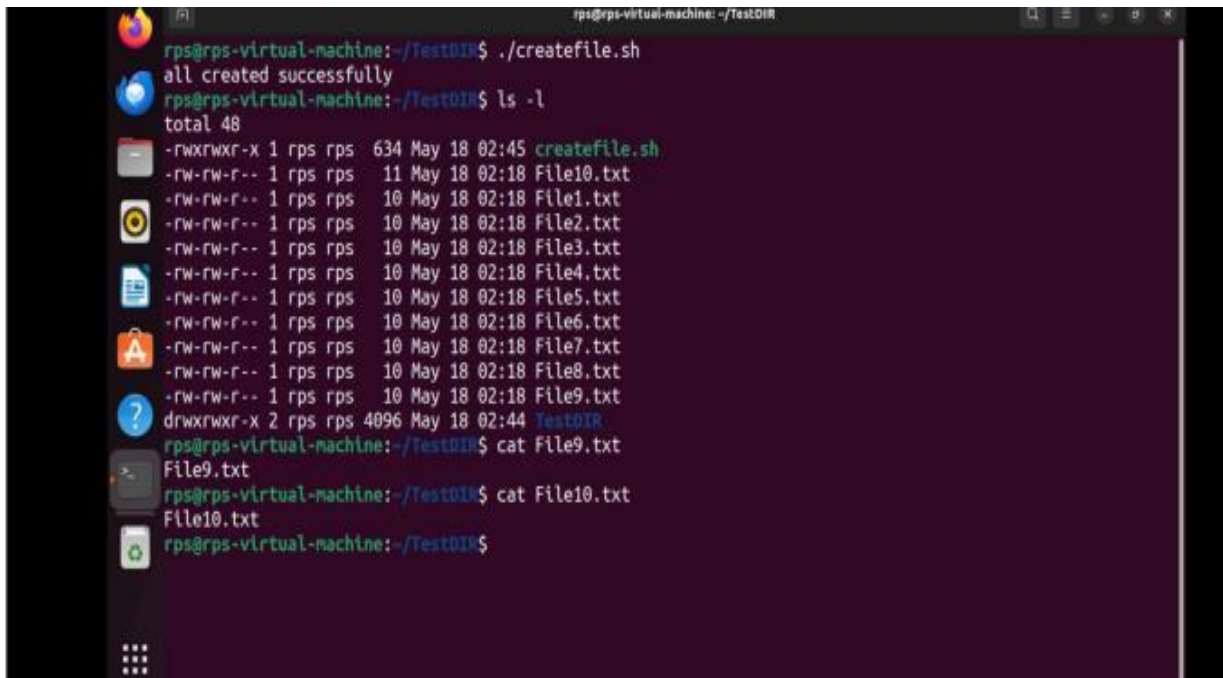
Assignment 5: Modify the script to handle errors, such as the directory already existing or lacking permissions to create files. Add a debugging mode that prints additional information when enabled.

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:



```
rps@rps-virtual-machine: ~/TestDIR
rps@rps-virtual-machine:~/TestDIR$ ./createfile.sh
all created successfully
rps@rps-virtual-machine:~/TestDIR$ ls -l
total 48
-rwxrwxr-x 1 rps rps 634 May 18 02:45 createfile.sh
-rw-rw-r-- 1 rps rps 11 May 18 02:18 File10.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File1.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File2.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File3.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File4.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File5.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File6.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File7.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File8.txt
-rw-rw-r-- 1 rps rps 10 May 18 02:18 File9.txt
drwxrwxr-x 2 rps rps 4096 May 18 02:44 TestDIR
rps@rps-virtual-machine:~/TestDIR$ cat File9.txt
File9.txt
rps@rps-virtual-machine:~/TestDIR$ cat File10.txt
File10.txt
rps@rps-virtual-machine:~/TestDIR$
```

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" |
```

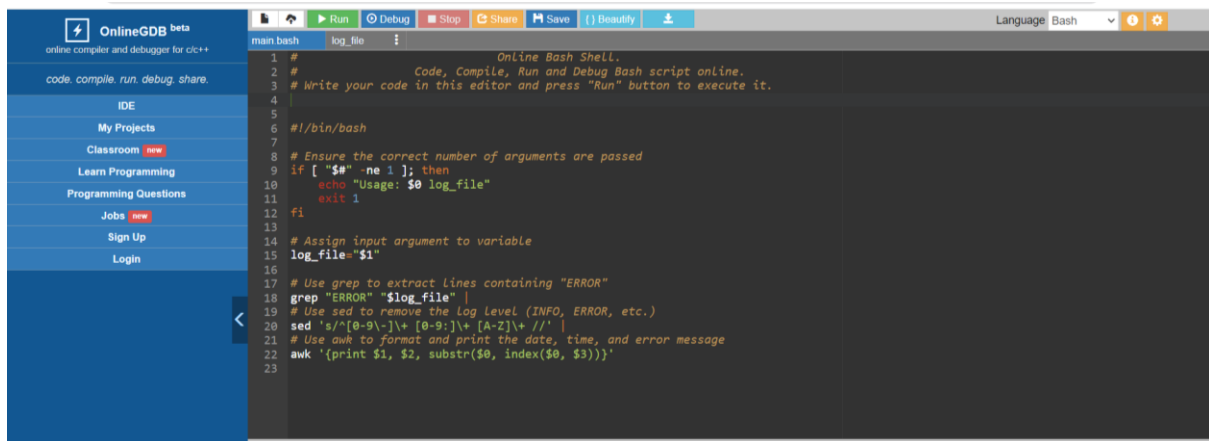
```
# Use sed to remove the log level (INFO, ERROR, etc.)
```

```
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

main.bash myfile.txt

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

```
1 #!/bin/bash
2
3 # Check if the correct number of arguments are provided
4 if [ $# -ne 3 ]; then
5     echo "Usage: $0 input_file old_text new_text"
6     exit 1
7 fi
8
9 # Assign input arguments to variables
10 input_file=$1
11 old_text=$2
12 new_text=$3
13 output_file="output_${input_file}"
14
15 # Check if the input file exists
16 if [ ! -f "$input_file" ]; then
17     echo "Error: Input file '$input_file' not found!"
18     exit 1
19 fi
20
21 # Use sed to replace all occurrences of old_text with new_text and save the result to a new file
22 sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"
23
24 # Output success message
25 echo "Replaced all occurrences of '$old_text' with '$new_text' in '$input_file' and saved to '$output_file'"
```

Language: Bash

Input