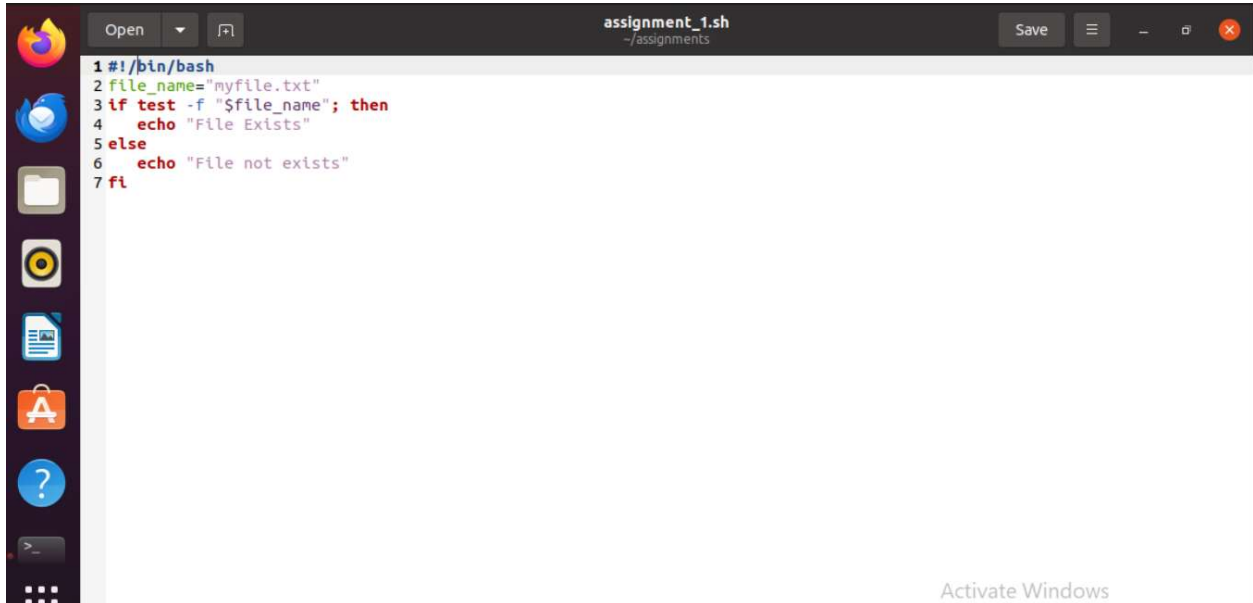


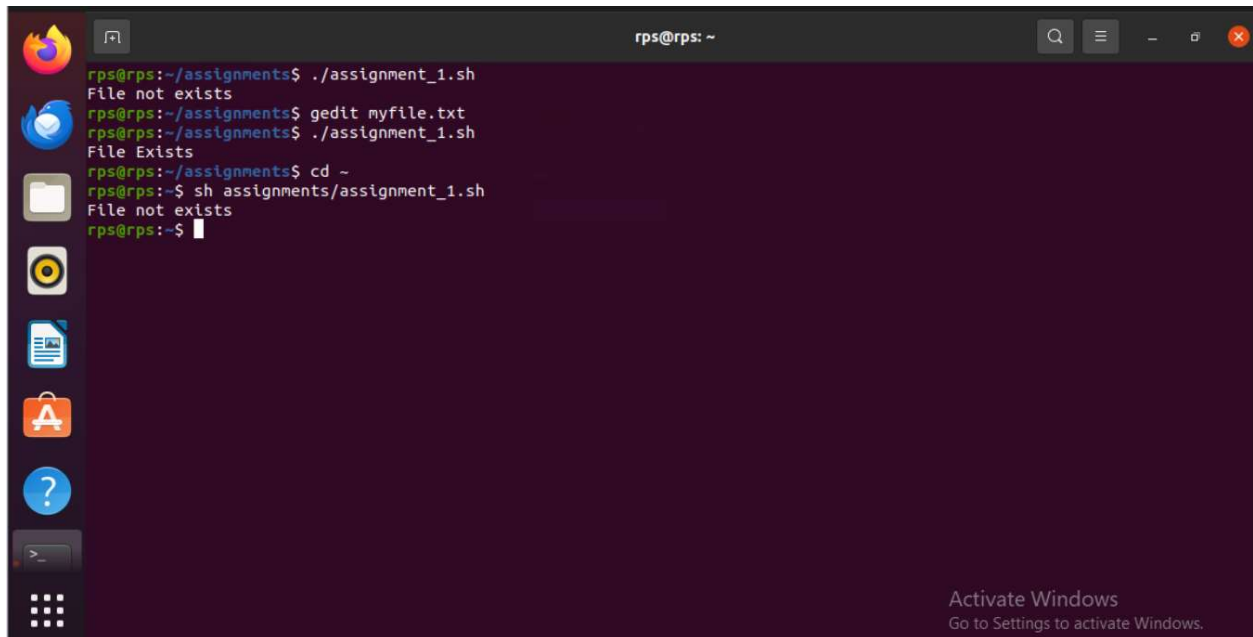
Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Shell script to check if a specific file (e.g., myfile.txt) exists in the current directory:

A screenshot of a code editor window titled "assignment_1.sh" with the path "~/assignments". The editor contains a shell script with seven lines of code. The left sidebar shows a dock with various application icons. The bottom right corner of the window has an "Activate Windows" watermark.

```
1#!/bin/bash
2file_name="myfile.txt"
3if test -f "$file_name"; then
4    echo "File Exists"
5else
6    echo "File not exists"
7fi
```

Terminal Output:

A screenshot of a terminal window with the prompt "rps@rps: ~". The terminal shows the execution of the script "assignment_1.sh" in three different contexts: first in the current directory where it outputs "File not exists", then after creating the file "myfile.txt" using "gedit", where it outputs "File Exists", and finally after navigating to the parent directory and running the script from there, where it outputs "File not exists". The left sidebar shows the same dock as the previous image. The bottom right corner has an "Activate Windows" watermark.

```
rps@rps:~/assignments$ ./assignment_1.sh
File not exists
rps@rps:~/assignments$ gedit myfile.txt
rps@rps:~/assignments$ ./assignment_1.sh
File Exists
rps@rps:~/assignments$ cd ~
rps@rps:~$ sh assignments/assignment_1.sh
File not exists
rps@rps:~$
```

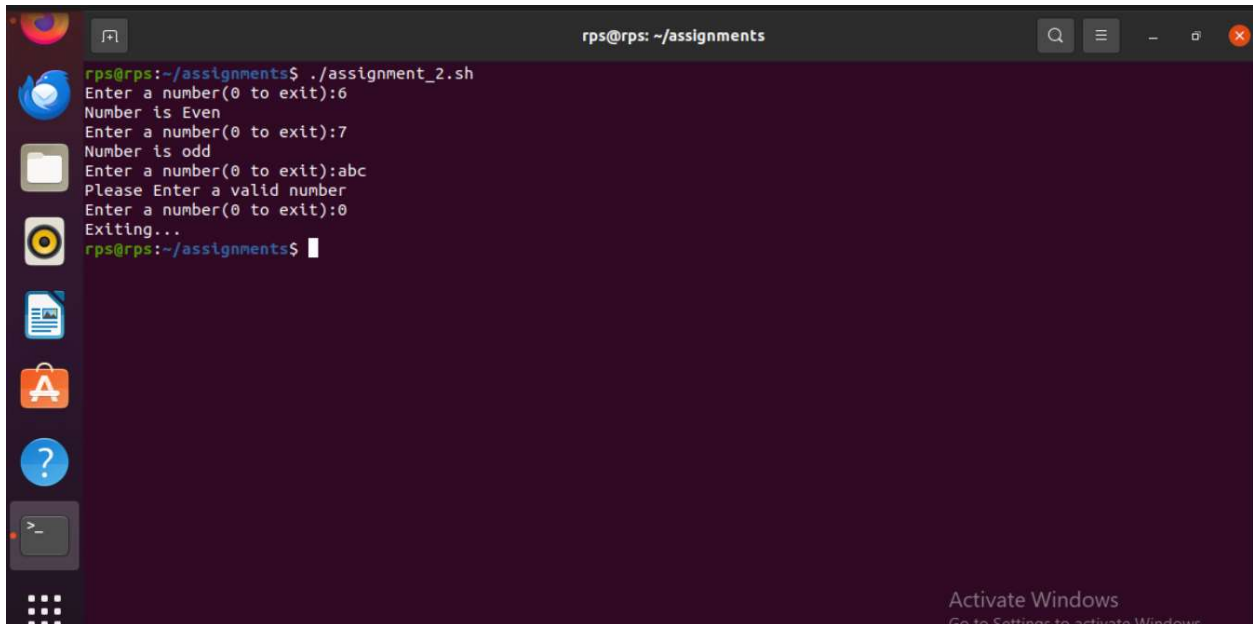
Assignment 2: Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.

Shell script to reads number and print whether each number is odd or even:

A screenshot of a code editor window titled "assignment_2.sh" with a dark theme. The editor shows a shell script with line numbers 1 through 28. The script defines two functions: check_fn() and read_fn(). The check_fn() function takes a number as input and prints "Number is Even" or "Number is odd" based on whether it is divisible by 2. The read_fn() function enters a loop where it prompts the user to enter a number (0 to exit), validates the input, and calls check_fn() if the input is a valid number. The script ends with a call to read_fn().

```
1#!/bin/bash
2check_fn(){
3    local num=$1
4
5    if((num % 2 == 0)); then
6        echo "Number is Even"
7    else
8        echo "Number is odd"
9    fi
10}
11
12read_fn(){
13    local number
14    while true; do
15        read -p "Enter a number(0 to exit):" number
16        if [[ $number =~ ^-?[0-9]+$ ]]; then
17            if test "$number" -eq 0; then
18                echo "Exiting..."
19                break
20            elif test "$number" -gt 0; then
21                check_fn "$number"
22            fi
23        else
24            echo "Please Enter a valid number"
25        fi
26    done
27}
28read_fn
```

Terminal Output:

A screenshot of a terminal window with a dark purple background. The prompt is "rps@rps: ~/assignments". The user has executed the command "./assignment_2.sh". The script's output is shown: it prompts for a number, and for inputs 6, 7, and 'abc', it prints "Number is Even", "Number is odd", and "Please Enter a valid number" respectively. Finally, for input 0, it prints "Exiting...".

```
rps@rps: ~/assignments
rps@rps:~/assignments$ ./assignment_2.sh
Enter a number(0 to exit):6
Number is Even
Enter a number(0 to exit):7
Number is odd
Enter a number(0 to exit):abc
Please Enter a valid number
Enter a number(0 to exit):0
Exiting...
rps@rps:~/assignments$
```

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.

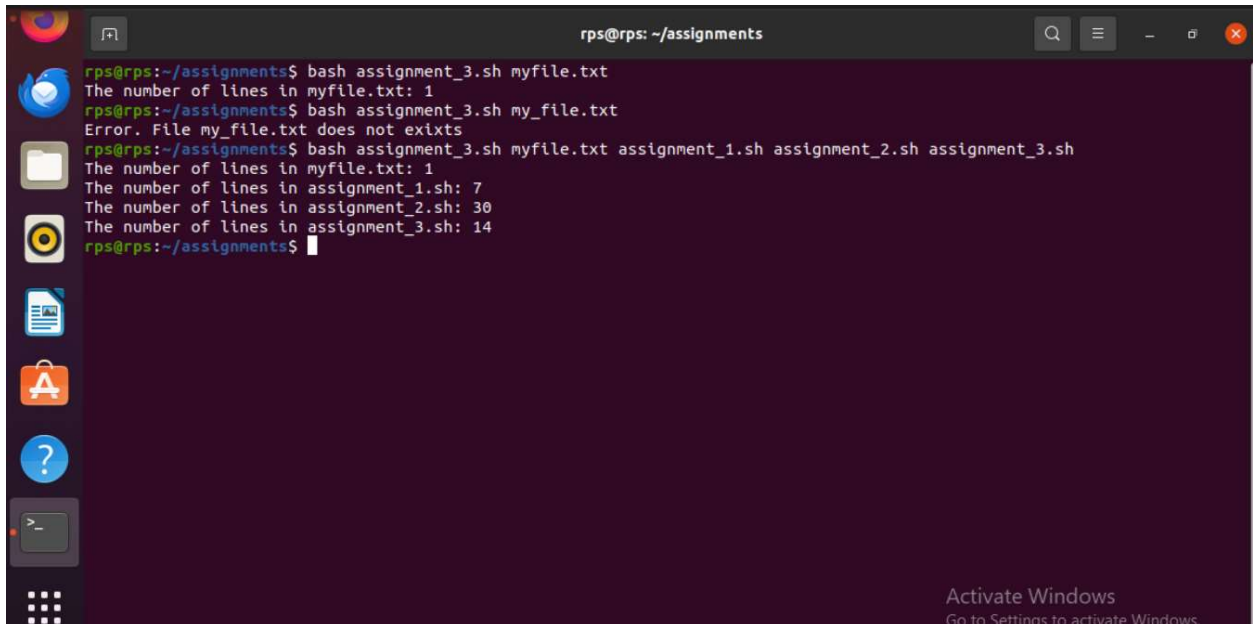
Shell script to write a function that takes a filename and prints the number of lines in the file:



```
1#!/bin/bash
2count_file_lines() {
3    local file_name=$1
4    line_count=$(wc -l < "$file_name")
5    echo "The number of lines in $file_name: $line_count"
6}
7
8for file_name in "$@"; do
9    if test -e "$file_name"; then
10        count_file_lines "$file_name"
11    else
12        echo "Error. File $file_name does not exists"
13    fi
14done
```

Activate Windows
Go to Settings to activate Windows.

Terminal Output:




```
rps@rps: ~/assignments
rps@rps:~/assignments$ bash assignment_3.sh myfile.txt
The number of lines in myfile.txt: 1
rps@rps:~/assignments$ bash assignment_3.sh my_file.txt
Error. File my_file.txt does not exists
rps@rps:~/assignments$ bash assignment_3.sh myfile.txt assignment_1.sh assignment_2.sh assignment_3.sh
The number of lines in myfile.txt: 1
The number of lines in assignment_1.sh: 7
The number of lines in assignment_2.sh: 30
The number of lines in assignment_3.sh: 14
rps@rps:~/assignments$
```

Activate Windows
Go to Settings to activate Windows.

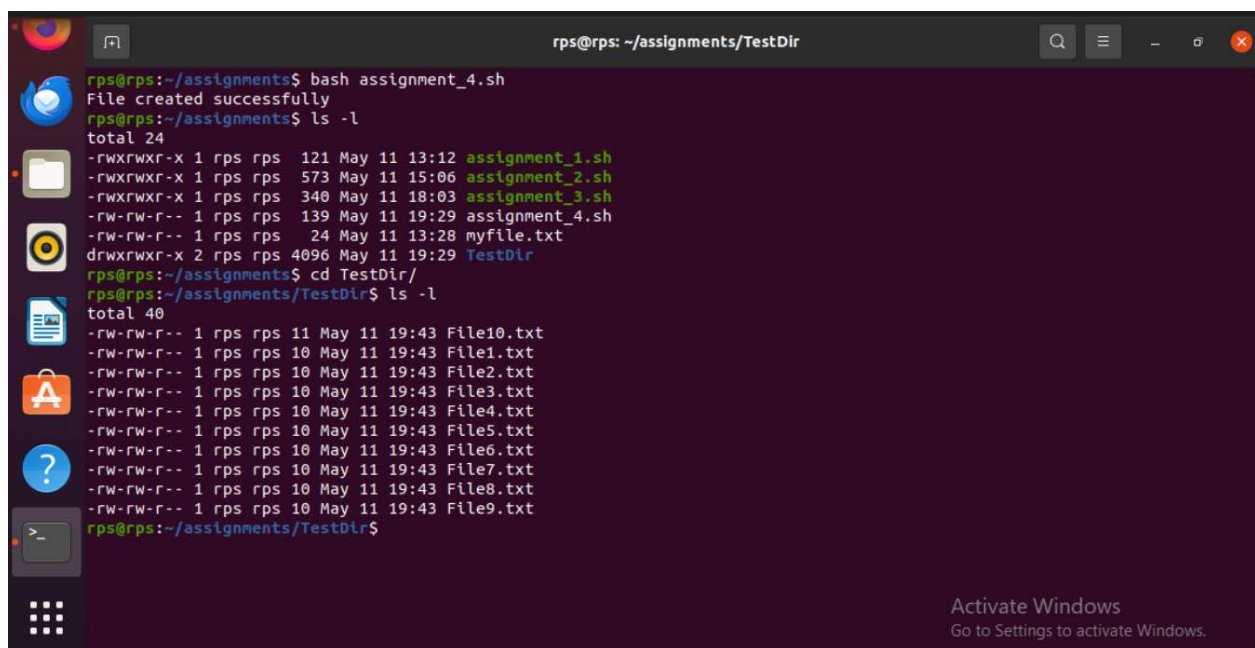
Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

Shell script to that creates a directory named TestDir and inside it creates ten files with its filename as its content:

A screenshot of a code editor window titled "assignment_4.sh" with a dark theme. The editor shows a shell script with four lines: 1. #!/bin/bash, 2. mkdir -p TestDir, 3. printf "File%s.txt\n" {1..10} | xargs -I {} sh -c 'echo "{}" > "TestDir/{}"', and 4. echo "File created successfully". The window has a sidebar on the left with various icons and a top bar with "Open", "Save", and window control buttons. At the bottom, there is a status bar showing "sh", "Tab Width: 8", "Ln 3, Col 69", and "INS". An "Activate Windows" watermark is visible in the bottom right corner.

```
1#!/bin/bash
2mkdir -p TestDir
3printf "File%s.txt\n" {1..10} | xargs -I {} sh -c 'echo "{}" > "TestDir/{}"'
4echo "File created successfully"
```

Terminal Output:

A screenshot of a terminal window titled "rps@rps: ~/assignments/TestDir" with a dark purple theme. It shows the execution of the script "assignment_4.sh" from the ~/assignments directory. The output shows "File created successfully", followed by a directory listing of ~/assignments showing the new TestDir and other files. Then, the user changes to the TestDir directory and lists its contents, showing ten files named File1.txt through File10.txt, each with its filename as content. The window has a sidebar on the left and a top bar with search, menu, and window control buttons. At the bottom, there is a status bar and an "Activate Windows" watermark.

```
rps@rps:~/assignments$ bash assignment_4.sh
File created successfully
rps@rps:~/assignments$ ls -l
total 24
-rwxrwxr-x 1 rps rps 121 May 11 13:12 assignment_1.sh
-rwxrwxr-x 1 rps rps 573 May 11 15:06 assignment_2.sh
-rwxrwxr-x 1 rps rps 340 May 11 18:03 assignment_3.sh
-rw-rw-r-- 1 rps rps 139 May 11 19:29 assignment_4.sh
-rw-rw-r-- 1 rps rps 24 May 11 13:28 myfile.txt
drwxrwxr-x 2 rps rps 4096 May 11 19:29 TestDir
rps@rps:~/assignments$ cd TestDir/
rps@rps:~/assignments/TestDir$ ls -l
total 40
-rw-rw-r-- 1 rps rps 11 May 11 19:43 File10.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File1.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File2.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File3.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File4.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File5.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File6.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File7.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File8.txt
-rw-rw-r-- 1 rps rps 10 May 11 19:43 File9.txt
rps@rps:~/assignments/TestDir$
```

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.

Shell script to handle errors with addition of debugging mode that prints additional information when enabled:

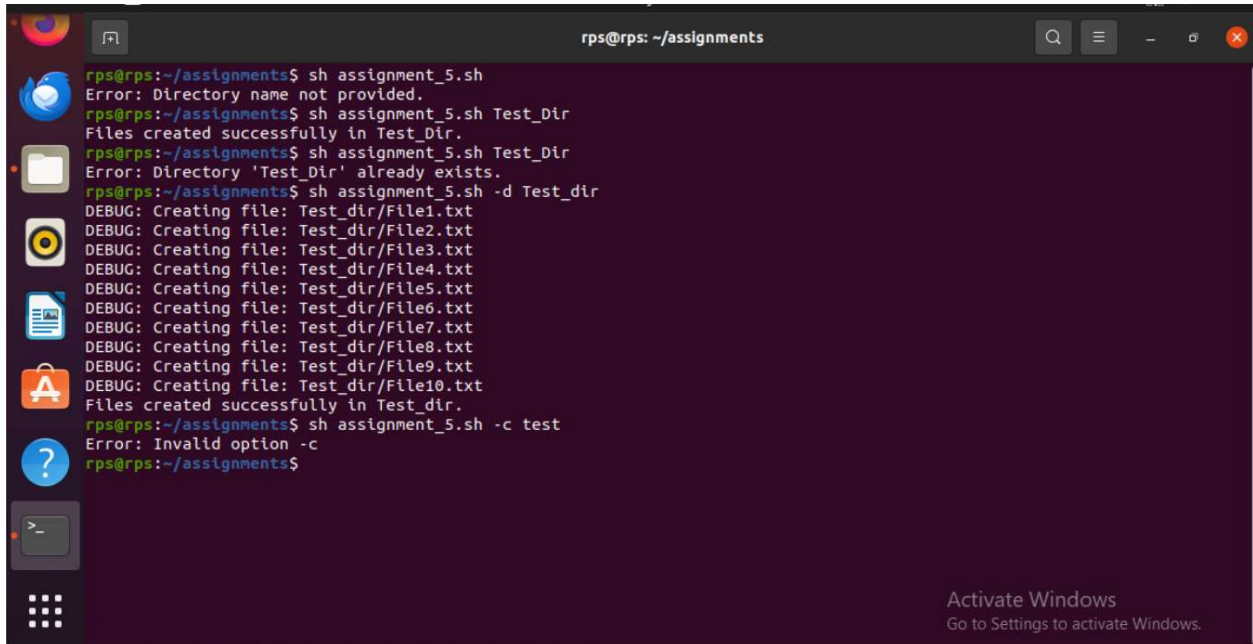
```
1 debug_msg() {
2   [ "$DEBUG" = "true" ] && echo "DEBUG: $1"
3 }
4
5 file_create() {
6   local directory="$1"
7   if [ -d "$directory" ]; then
8     echo "Error: Directory '$directory' already exists." >&2
9     exit 1
10  fi
11
12  mkdir -p "$directory" || { echo "Error: Failed to create directory '$directory'." >&2; exit 1; }
13
14  filename=""
15  for i in $(seq 1 10); do
16    filename="$directory/File${i}.txt"
17    debug_msg "Creating file: $filename"
18    echo "File${i}.txt" > "$filename" || { echo "Error: Failed to create file '$filename'." >&2; exit 1; }
19  done
20
21  echo "Files created successfully in $directory."
22 }
23
24 while getopts ":d" opt; do
25   case $opt in
26     d) DEBUG="true";;
27     \?) echo "Error: Invalid option -$OPTARG" >&2; exit 1;;
```

```
28   esac
29 done
30
31 shift $((OPTIND -1))
32
33 if [ $# -eq 0 ]; then
34   echo "Error: Directory name not provided." >&2
35   exit 1
36 fi
37
38 file_create "$1"
39
```

sh Tab Width: 8 Ln 35, Col 10 INS

Activate Windows

Terminal Output:



```
rps@rps: ~/assignments
rps@rps:~/assignments$ sh assignment_5.sh
Error: Directory name not provided.
rps@rps:~/assignments$ sh assignment_5.sh Test_Dir
Files created successfully in Test_Dir.
rps@rps:~/assignments$ sh assignment_5.sh Test_Dir
Error: Directory 'Test_Dir' already exists.
rps@rps:~/assignments$ sh assignment_5.sh -d Test_dir
DEBUG: Creating file: Test_dir/File1.txt
DEBUG: Creating file: Test_dir/File2.txt
DEBUG: Creating file: Test_dir/File3.txt
DEBUG: Creating file: Test_dir/File4.txt
DEBUG: Creating file: Test_dir/File5.txt
DEBUG: Creating file: Test_dir/File6.txt
DEBUG: Creating file: Test_dir/File7.txt
DEBUG: Creating file: Test_dir/File8.txt
DEBUG: Creating file: Test_dir/File9.txt
DEBUG: Creating file: Test_dir/File10.txt
Files created successfully in Test_dir.
rps@rps:~/assignments$ sh assignment_5.sh -c test
Error: Invalid option -c
rps@rps:~/assignments$
```

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

Shell script to extract all lines containing "ERROR" and to print the date, time, and error message:



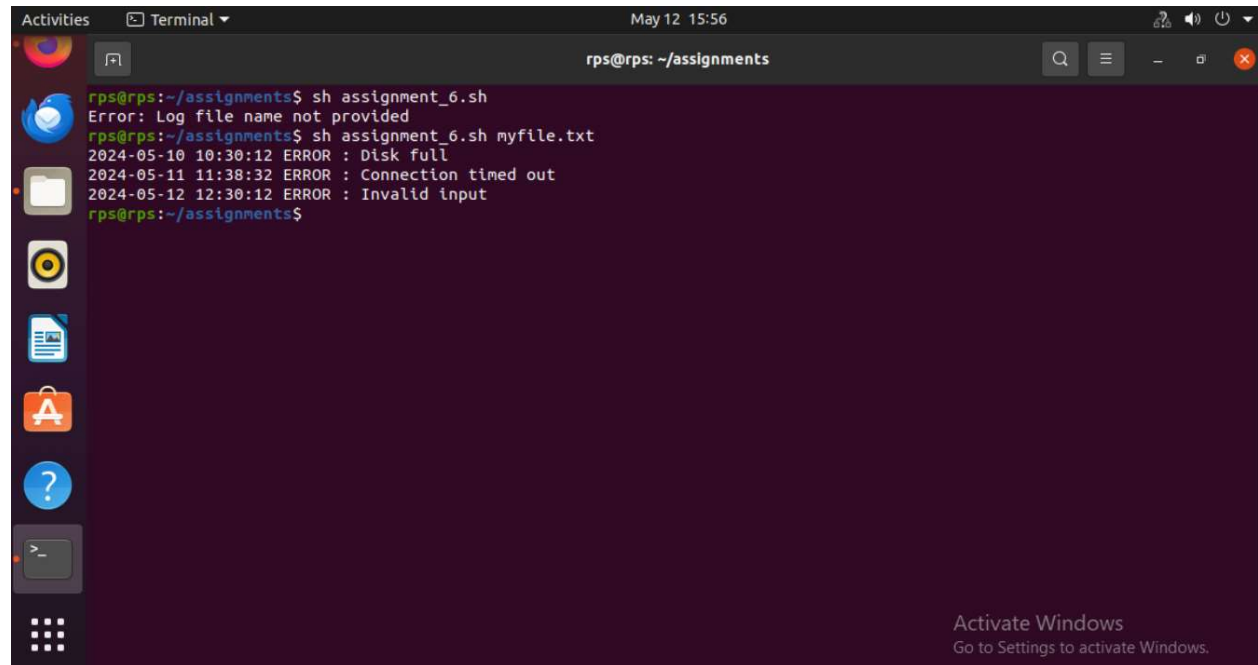
```
1#!/bin/bash
2file_exists() {
3    local file="$1"
4    [ -f "$file" ]
5}
6
7print_error() {
8    local msg="$1"
9    echo "Error: $msg">&2
10   exit 1
11}
12
13extract_error() {
14    local logfile="$1"
15    grep "ERROR" "$logfile" | awk '{print $1, $2, substr($0, index($0,$3))}' | sed 's/[\\/] / /; s/[0-9]
16}
17
18main() {
19    if [ $# -eq 0 ]; then
20        print_error "Log file name not provided"
21    fi
22
23    local logfile="$1"
24
25    if ! file_exists "$logfile"; then
26        print_error "Log file '$logfile' not found or is not a regular file"
27    fi
28}
```

```
28
29   extract_error "$logfile"
30 }
31
32 main "$@"
```

Activate Windows

Go to Settings to activate Windows.

Terminal Output:

A terminal window titled 'Terminal' with a dark background. The window shows the user 'rps' at the prompt 'rps@rps: ~/assignments'. The user runs 'sh assignment_6.sh', which results in an error: 'Error: Log file name not provided'. Then, the user runs 'sh assignment_6.sh myfile.txt', which produces three lines of error messages: '2024-05-10 10:30:12 ERROR : Disk full', '2024-05-11 11:38:32 ERROR : Connection timed out', and '2024-05-12 12:30:12 ERROR : Invalid input'. The terminal window also features a sidebar with application icons and a top status bar showing the date and time as 'May 12 15:56'.

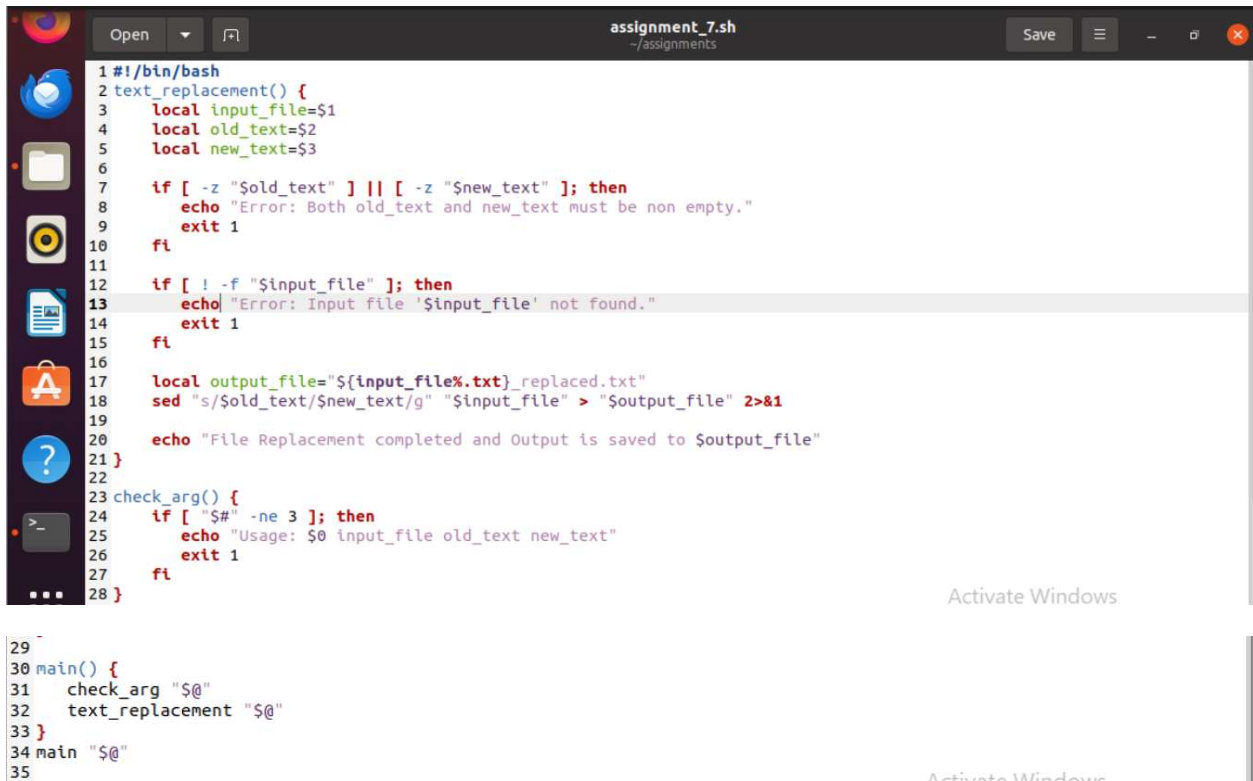
```
Activities Terminal May 12 15:56
rps@rps: ~/assignments
rps@rps:~/assignments$ sh assignment_6.sh
Error: Log file name not provided
rps@rps:~/assignments$ sh assignment_6.sh myfile.txt
2024-05-10 10:30:12 ERROR : Disk full
2024-05-11 11:38:32 ERROR : Connection timed out
2024-05-12 12:30:12 ERROR : Invalid input
rps@rps:~/assignments$
```

Activate Windows

Go to Settings to activate Windows.

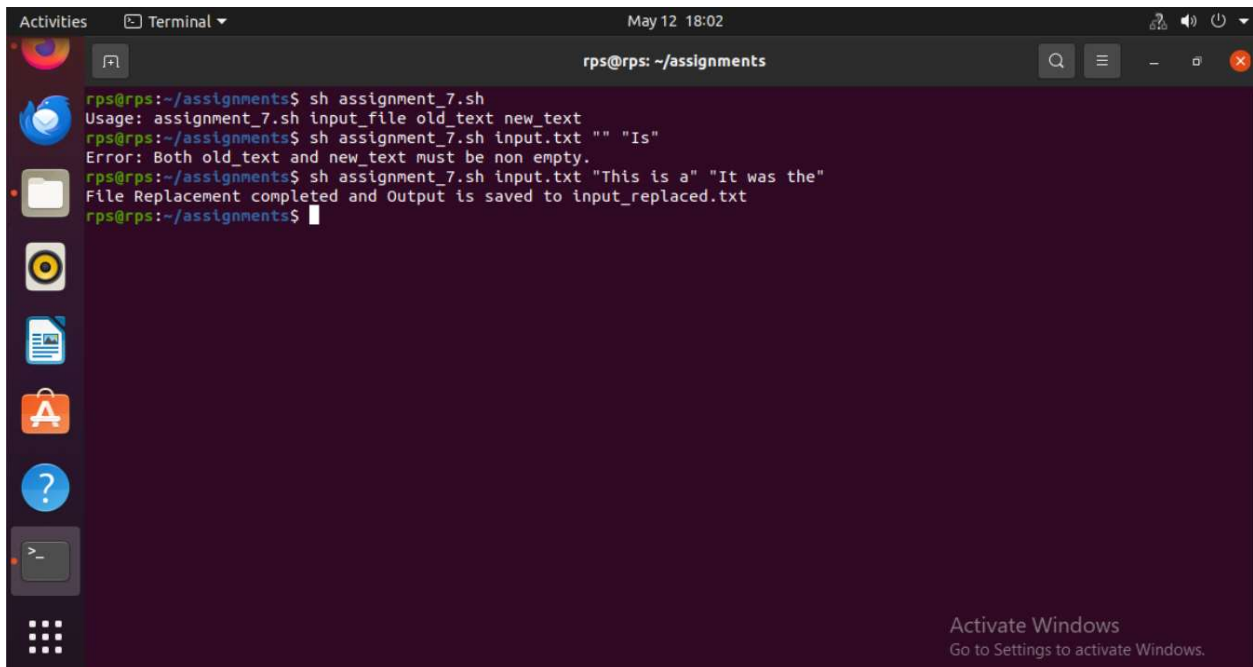
Assignment 7: Create a script that takes a text file and replaces all occurrences of "old_text" with "new_text". Use sed to perform this operation and output the result to a new file.

Shell script to take a text file and replaces all occurrences of "old_text" with "new_text":

A screenshot of a code editor window titled "assignment_7.sh" with the path "~/assignments". The editor shows a shell script with line numbers 1 through 35. The script defines a function "text_replacement" and a "main" function. The "text_replacement" function checks for non-empty arguments and the existence of the input file before using "sed" to replace "old_text" with "new_text" in the input file, saving the result to a new file. The "main" function calls "check_arg" and "text_replacement".

```
1#!/bin/bash
2text_replacement() {
3    local input_file=$1
4    local old_text=$2
5    local new_text=$3
6
7    if [ -z "$old_text" ] || [ -z "$new_text" ]; then
8        echo "Error: Both old_text and new_text must be non empty."
9        exit 1
10    fi
11
12    if [ ! -f "$input_file" ]; then
13        echo "Error: Input file '$input_file' not found."
14        exit 1
15    fi
16
17    local output_file="${input_file%.txt}_replaced.txt"
18    sed "s/$old_text/$new_text/g" "$input_file" > "$output_file" 2>&1
19
20    echo "File Replacement completed and Output is saved to $output_file"
21 }
22
23 check_arg() {
24     if [ "$#" -ne 3 ]; then
25         echo "Usage: $0 input_file old_text new_text"
26         exit 1
27     fi
28 }
29
30 main() {
31     check_arg "$@"
32     text_replacement "$@"
33 }
34 main "$@"
35
```

Terminal Output:

A screenshot of a terminal window titled "rps@rps: ~/assignments" with the date and time "May 12 18:02". The terminal shows the execution of the script "assignment_7.sh" with various arguments. It displays the usage message, an error for missing arguments, and a successful execution where "input.txt" is replaced with "input_replaced.txt".

```
rps@rps:~/assignments$ sh assignment_7.sh
Usage: assignment_7.sh input_file old_text new_text
rps@rps:~/assignments$ sh assignment_7.sh input.txt "" "Is"
Error: Both old_text and new_text must be non empty.
rps@rps:~/assignments$ sh assignment_7.sh input.txt "This is a" "It was the"
File Replacement completed and Output is saved to input_replaced.txt
rps@rps:~/assignments$
```