# Shell Scripting With Bash Assignment

## 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) vi ssl.sh

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

2) chmod 777 ssl.sh ---- Granting permissions
3) ./ssl.sh ---- running the script

## 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 (enter 0 to exit): " num
  if [ $num -eq 0 ]; then
    echo "Exiting the script..."
```

```
        break
    fi
    if [ $((num % 2)) -eq 0 ]; then
        echo "$num is even."
    else
        echo "$num is odd."
    fi
done
```

- ➢ odd_even.sh ------- Saving the script in a file
- ➢ chmod 777 odd_even.sh -------- Granting permissions
- ➢ ./odd_even.sh --------- Running 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.

1. creating the script

```
vi line.sh
```

2. Adding the code to the script

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

3. Save the script and grant the permissions by using this command chmod 777 line.sh
4. Run the script using ./line.sh

## 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
# Creating the TestDir directory
mkdir TestDir
# Change to the TestDir directory
cd TestDir
# Create the files and write the filename as content
for i in {1..10}; do
    filename="File$i.txt"
    echo "$filename" > "$filename"
done
```

```
# List the files in the directory
ls -l
```

## 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=false
# Function to display debug information
debug() {
   if [ "$DEBUG" = true ]; then
      echo "Debug: $1"
   fi
}

# Check if TestDir already exists
if [ -d "TestDir" ]; then
   echo "Error: Directory 'TestDir' already exists."
   exit 1
fi

# Create the TestDir directory
mkdir TestDir
```

```
if [ $? -ne 0 ]; then
   echo "Error: Unable to create directory 'TestDir'. Check permissions."
   exit 1
fi

# Change to the TestDir directory
cd TestDir

# Create the files and write the filename as content
for i in {1..10}; do
   filename="File$i.txt"
   debug "Creating file: $filename"
   echo "$filename" > "$filename"
   if [ $? -ne 0 ]; then
      echo "Error: Unable to create file '$filename'. Check permissions."
      exit 1
   fi
done

# List the files in the directory
ls -l
```

## 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

```
#!/bin/bash

# Set the log file path

log_file="path/to/your/logfile.log"

# Extract lines containing "ERROR" using grep

error_lines=$(grep "ERROR" "$log_file")

# Process the extracted lines using awk

awk -F'|' '{

    date = substr($1, 1, 10)

    time = substr($1, 12, 8)

    error_msg = $2

    print "Date: " date ", Time: " time ", Error: " error_msg

}' <<< "$error_lines"
```

## 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 is 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="output.txt"

# Perform the text replacement using sed and save the output to a new file

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

echo "Text replacement completed. Output saved to $output_file."
```

- Save the script to a file (e.g., replace_text.sh).
- Make the script executable using chmod +x replace_text.sh.
- Run the script with the input file path, old text, and new text as arguments. For example:
- ./replace_text.sh input.txt old_text new_text
- After running the script, it will replace all occurrences of "old_text" with "new_text" in the input file and save the modified content to a new file named output.txt.