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

# Function to display debug messages

debug_msg() {
   if [ "$DEBUG" = true ]; then
      echo "[DEBUG] $1"
   fi
}


# Function to create directory and files

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


   # Attempt to create directory TestDir
   mkdir -p TestDir
   if [ $? -ne 0 ]; then
```

```
        echo "Error: Unable to create directory TestDir."
        exit 1
    fi

    debug_msg "Directory TestDir created successfully."

    # Loop to create ten files inside TestDir
    for ((i=1; i<=10; i++)); do
        filename="TestDir/File$i.txt"
        debug_msg "Creating $filename"
        echo "File$i.txt" > "$filename"
        if [ $? -ne 0 ]; then
            echo "Error: Unable to create file $filename."
            exit 1
        fi
    done
    echo "Files created successfully."
}
# Main script starts here
# Parse command-line arguments
while getopts ":d" opt; do
    case ${opt} in
```

```
    d )
      DEBUG=true
      ;;
    \? )
      echo "Usage: $0 [-d] (enable debugging)"
      exit 1
      ;;
  esac
done

# Call function to create files
create_files
exit 0
```

This script introduces a debug_msg function to display debug messages when the debugging mode is enabled (-d option). Error handling is implemented using conditional checks after critical commands such as mkdir and file creation. Additionally, it checks for the existence of the directory before attempting to create it.

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

# Using grep to extract lines containing "ERROR"

grep "ERROR" sample.log | \

# Using awk to print date, time, and error message

awk '{print $1, $2, $3, $0}' | \

# Using sed to format the output

sed -E 's/^([A-Za-z]+ [0-9]+ [0-9]+:[0-9]+:[0-9]+).*ERROR: (.*)$/Date: \1, Error: \2/'
```

This script does the following:

grep "ERROR" sample.log: Searches for lines containing "ERROR" in the file sample.log.

awk '{print $1, $2, $3, $0}': Prints the first three fields (date, time) along with the entire line. This assumes that the date and time are in the first three fields of each line.

sed -E 's/^([A-Za-z]+ [0-9]+ [0-9]+:[0-9]+:[0-9]+).*ERROR: (.*)$/Date: \1, Error: \2/': Uses sed to format the output by capturing the date and

time followed by "ERROR: ", and then capturing the error message. It then prints the formatted date, time, and error message.

You can save this script to a file, let's say extract_errors.sh, give it execute permissions using chmod +x extract_errors.sh, and then run it using ./extract_errors.sh. It will extract lines containing "ERROR" from sample.log and print the formatted 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.

```
#!/bin/bash

# Check if 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%.txt}_modified.txt"
```

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

# Perform text replacement using sed and save output to new file
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

echo "Replacement completed. Result saved to $output_file."
```

You can save this script to a file, let's say replace_text.sh, give it execute permissions using chmod +x replace_text.sh, and then run it using ./replace_text.sh input_file.txt old_text new_text.

Replace input_file.txt with the path to your text file, old_text with the text you want to replace, and new_text with the text you want to replace it with. The modified content will be saved to a new file named input_file_modified.txt.