**Assignment on 02-08-2024**

**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".

#!/bin/bash

# Define the filename

filename="myfile.txt"

# Check if the file exists

if [ -f "$filename" ]; then

echo "File exists"

else

echo "File not found"

fi

Here's a breakdown of the script:

1. **#!/bin/bash:** This line specifies that the script should be run using the Bash shell.
2. **filename="myfile.txt":** This sets the variable filename to the name of the file you're checking for.
3. **if [ -f "$filename" ]; then**: This checks if the file exists and is a regular file. The -f flag checks for regular files (not directories).
4. **echo "File exists":** This line is executed if the file is found.
5. **else:** This starts the block of code that will be executed if the file is not found.
6. **echo "File not found":** This line is executed if the file is not found.

**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

# Prompt the user to enter a number

echo -n "Enter a number (0 to exit): "

read number

# Check if the user entered '0'

if [ "$number" -eq 0 ]; then

echo "Exiting..."

break

fi

# Check if the number is odd or even

if [ $((number % 2)) -eq 0 ]; then

echo "$number is even"

else

echo "$number is odd"

fi

done

**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.

#!/bin/bash

# Function to count the number of lines in a file

count\_lines() {

local filename=$1

# Check if the file exists

if [ -f "$filename" ]; then

# Count the number of lines and print the result

local line\_count=$(wc -l < "$filename")

echo "The file '$filename' has $line\_count lines."

else

echo "The file '$filename' does not exist."

fi

}

# Call the function with different filenames

count\_lines "myfile1.txt"

count\_lines "myfile2.txt"

count\_lines "myfile3.txt"

**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

# Create a directory named TestDir

mkdir -p TestDir

# Navigate into the directory

cd TestDir

# Create 10 files and write their names into them

for i in {1..10}; do

filename="File$i.txt"

echo "$filename" > "$filename"

done

**Assignment 5:** Modify the script to handle errors, such as the directory already existing or lacking permissions to create files.

#!/bin/bash

# Define the directory name

dir\_name="TestDir"

# Create a directory named TestDir if it doesn't exist

if [ ! -d "$dir\_name" ]; then

mkdir "$dir\_name" 2>/dev/null

if [ $? -ne 0 ]; then

echo "Error: Failed to create directory '$dir\_name'. You may not have the necessary permissions."

exit 1

fi

else

echo "Warning: Directory '$dir\_name' already exists."

fi

# Navigate into the directory

cd "$dir\_name" 2>/dev/null

if [ $? -ne 0 ]; then

echo "Error: Failed to navigate to directory '$dir\_name'."

exit 1

fi

# Create 10 files and write their names into them

for i in {1..10}; do

filename="File$i.txt"

echo "$filename" > "$filename" 2>/dev/null

if [ $? -ne 0 ]; then

echo "Error: Failed to create or write to file '$filename'."

else

echo "File '$filename' created successfully."

fi

done

**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.

#!/bin/bash

# Define the log file name

log\_file="sample.log"

# Use grep to extract lines containing "ERROR" and awk to process the output

grep "ERROR" "$log\_file" | awk '{print $1, $2, $3}'

2024-08-02 10:15:30 INFO Starting the process

2024-08-02 10:15:31 ERROR Failed to connect to the database

2024-08-02 10:15:32 INFO Process running

2024-08-02 10:15:33 ERROR Unexpected input format

2024-08-02 10:15:31 ERROR Failed to connect to the database

2024-08-02 10:15:33 ERROR Unexpected input format

The script extracts lines containing "ERROR" and prints the date, time, and error message for each. If your log file format differs, you may need to adjust the awk field extraction accordingly.

**Assignment 7:** Create a script that takes a text file and replaces all occurrences of " ld t t" ith " t t" U d t f thi ti d t t th lt t \_x000D\_# Confidential - RPS Data 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

# Assign input arguments to variables

input\_file="$1"

old\_text="$2"

new\_text="$3"

# Define the output file name

output\_file="modified\_$(basename "$input\_file")"

# Use sed to replace all occurrences of old\_text with new\_text and write to the output file

sed "s/$old\_text/$new\_text/g" "$input\_file" > "$output\_file"

# Print a message indicating the output file

echo "Replaced '$old\_text' with '$new\_text' in '$input\_file' and saved the result to '$output\_file'."