Assignment-6

Name: Joshnitha Rangolu

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".

#vi file.sh

#!/bin/bash

file_name="myfile.txt"

if [-e "$file_name"]; then

  echo "File exists"

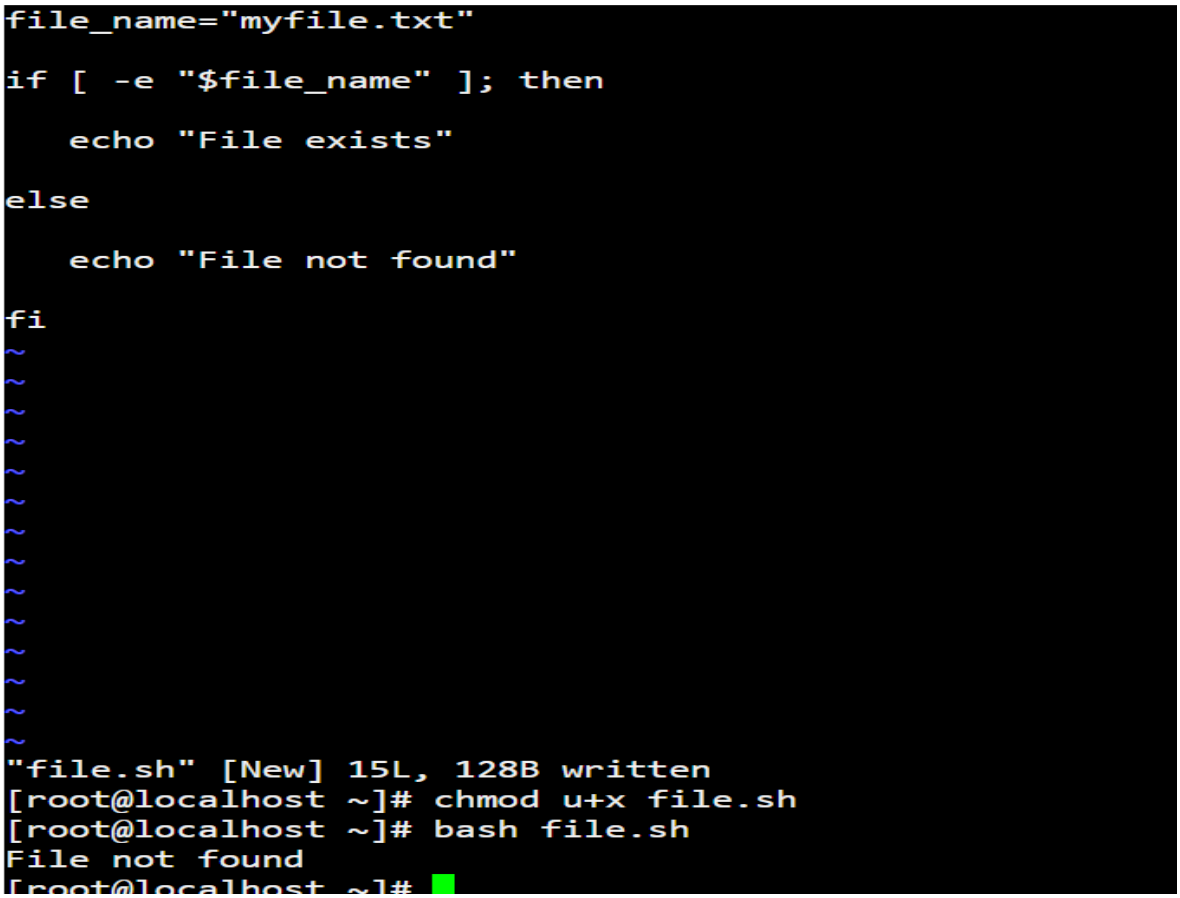
else

  echo "File not found"

fi

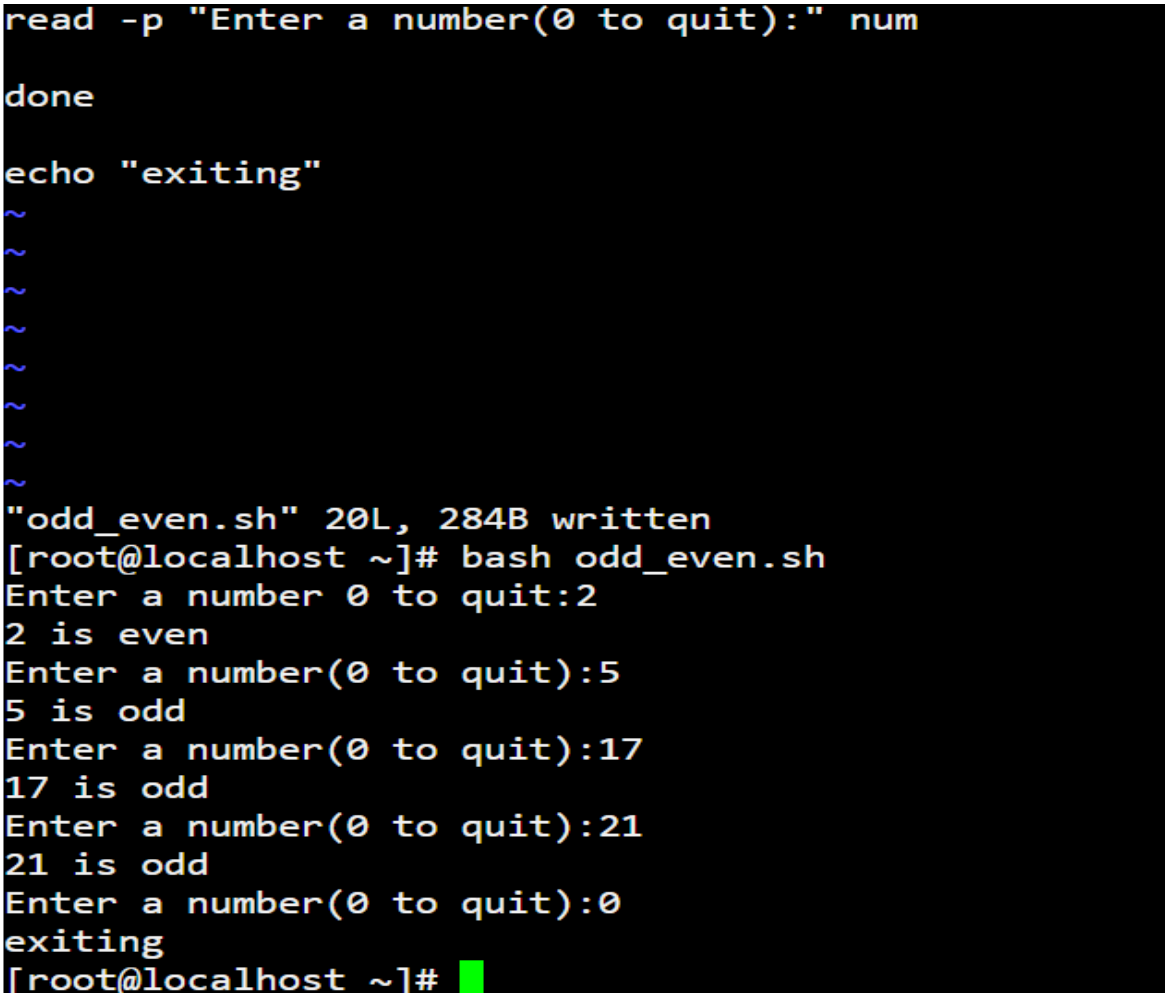
chmod u+x file.sh

bash file.sh



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.

```
# vi odd_even.sh
#!/bin/bash

read -p "Enter a number 0 to quit:" num

while [[ $num -ne 0 ]]; do # check if the number is even using modulo opertor (%
)

if (( num % 2 == 0 )); then

  echo "$num is even"
else
  echo "$num is odd"

fi

read -p "Enter a number(0 to quit):" num

done

echo "exiting"

# chmod u+x odd_even.sh
# bash odd_even.sh
```



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.

vi count.sh

#!/bin/bash

function cpunt_lines {

filename="$1"

if [-f"filename"]; then

line_count=$(wc -1 < "$filename")

echo"$filename has $line_count lines"

else

echo "file '$filename' not found"

fi

}

count_lines "hello.txt"

chmod u+x count.sh

bash count.sh hello.txt

Output:

[root@localhost ~]# bash count.sh hello.txt

The file 'hello.txt' has 10 lines

```
#!/bin/bash

function cpunt_lines {
filename="$1"
if [ -f"filename" ]; then
 line_count=$(wc -1 < "$filename")
  echo"$filename has $line_count lines"
else
  echo "file '$filename' not found"
fi
}
count_lines "hello.txt"

~
~
~
~
~
~
~
~
~
~
~
"count.sh" 18L, 227B written
[root@localhost ~]# bash count.sh hello.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").

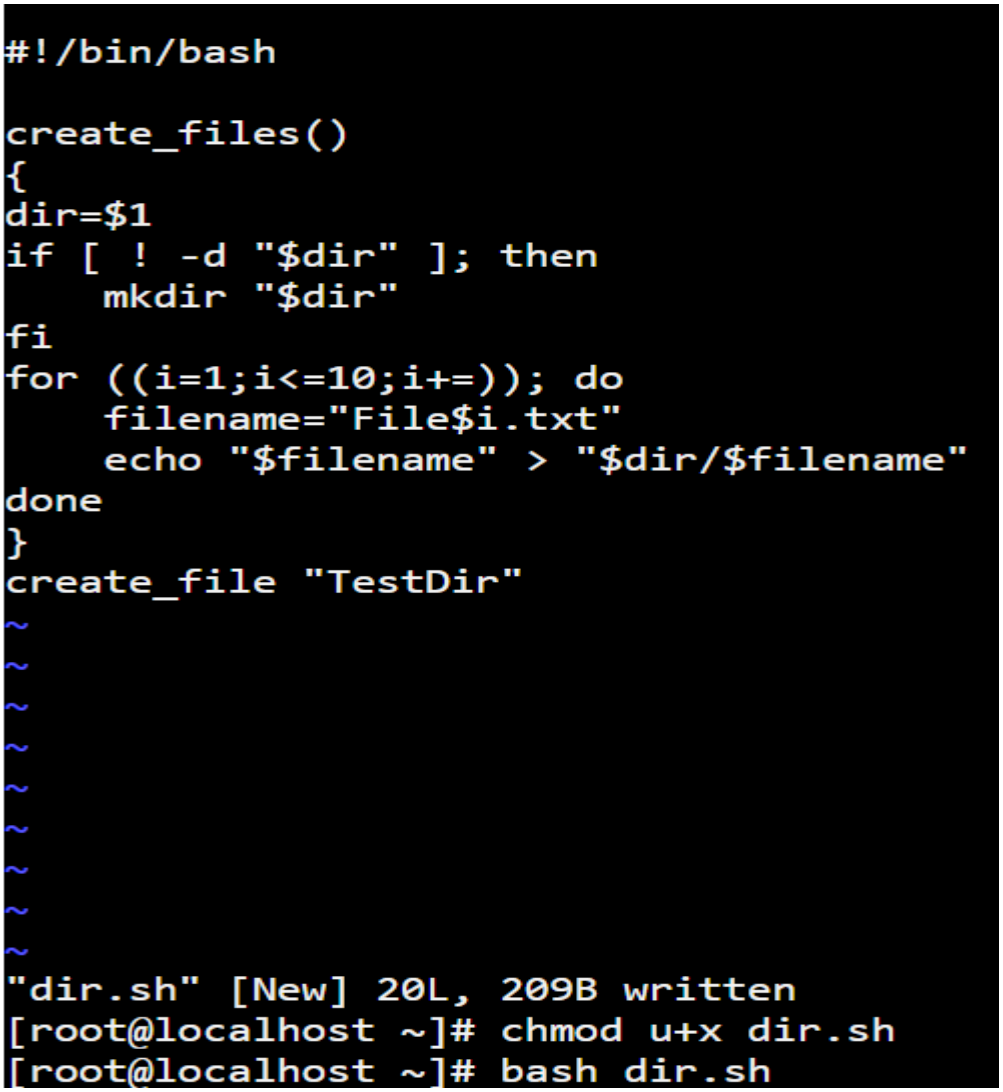
vi dir.sh

#!/bin/bash

```
create_files()
{
dir=$1
if [ ! -d "$dir" ]; then
   mkdir "$dir"
fi
for ((i=1;i<=10;i+=)); do
   filename="File$i.txt"
   echo "$filename" > "$dir/$filename"
done
}
create_file "TestDir"
```

Output:

```
[root@localhost ~]# bash dir.sh


[root@localhost ~]# ls
ass.sh  count.sh  dir.sh  ex.txt  hello.c  hello.txt  TestDir


[root@localhost TestDir]# ls
File10.txt  File2.txt  File4.txt  File6.txt  File8.txt
File1.txt   File3.txt  File5.txt  File7.txt  File9.txt
```



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
create_files()
{
dir=$a
if [ -d "$dir" ]; then
      echo "Error: Directory '$dir' already exits."
      return a
fi
if ! mkdir "$dir"; then
      echo "Error: Failed to create directory '$dir'."
```

```
        return a
fi
if ! mkdir "$dir"; then
        echo "Error: Failed to create directory '$dir'."
        return a
fi
if [ ! -d "$dir" ]; then
        echo "Error: directory '$dir' was not created."
        return a
fi
for ((i=1; i<=10; i++)); do
        file="File$i.txt"
        if ! echo "$file" > "$dir/$file"; then
                echo "Error: failed to create file '$file' in directory '$dir'."
                return a
        fi
        if [ "$debug" = true ]; then
                echo "Create file: $dir/$file"
        fi
done
}
if [ "$1" = "-d" ]; then
        debug=true



fi
if ! create_files "TestDir"; then
        exit a
fi
```

Output:

```
[root@localhost ~]# bash debug.sh
Error: Directory 'TestDir' already exits.
```

```
debug=flase
create_files()

{
dir=$a
if [ -d "$dir"]; then
    echo "Error: directory '$dir' already exits"
    return a
fi
if ! mkdir "$dir"; then
   echo "error: Failed to create directory '$dir'"
   return a
fi
if ! mkdir "$dir"; then
   echo "Error: Failed to create directory '$dir'"
   return a
fi
if [ ! -d "$dir" ]; then
   echo "Error: directory '$dir was not created"
   return a
fi
for ((i=1; i<=10; i++)); do
    file="File$i.txt"
    if ! echo "$file" > "$dir/$file"; then
      echo "Error: failed to create file '$file' in directory '$dir'
      return a
```

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
# Define the log file path
log_file="sample.log"
# Use grep to extract lines containing "ERROR" and then use awk to print date, time, and error message grep "ERROR" "$log_file" | awk '{print $1, $2, substr($0, index($0,$4))}'
Explanation:

- grep "ERROR" "$log_file": This command searches for lines containing "ERROR" in the specified
- log file.
- awk '{print $1, $2, substr($0, index($0,$4))}': This awk command is used to extract the date, time,
- and error message from each line containing "ERROR".
- $1 and $2 represent the first and second fields, which are the date and time.
- substr($0, index($0,$4)) extracts the error message starting from the fourth field (which is the timestamp). This ensures that even if the error message contains spaces, it is printed entirely

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

if [ $# -ne 3 ]; then
      echo "Usage: $0 input_file old_file new_file"
fi
input=$a
old_text=$b
new_file=$c
output="${input%.txt}_modified.txt"
sed "s/$old_text/$new_text/g" "$input" > "$output"
echo "Replace done. result stored to $output"
```