Assignment – 6

(Shell Scripting)

Prakash Manikanta Irrinki
Prakashnaidu9494@gmail.com

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".

Code:

```
file="myfile.txt"
if [ -e "$file" ]; then
echo "File exists"
else
echo "File not found"
fi
```

output:

```
[root@localhost assignments]# bash if.sh
File exists
```

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.

Code:

```
while :
do
echo "Enter a number(enter 0 to stop)"
read num
if [ $num -eq 0 ]
then
    exit
fi
if [ `expr $num % 2` -eq 0 ]
then
    echo "$num is even"
else
    echo "$num is odd"
fi
done
```

Output:

```
[root@localhost ~]# bash ass.sh
Enter a number(enter 0 to stop)
2
2 is even
Enter a number(enter 0 to stop)
7
7 is odd
Enter a number(enter 0 to stop)
24
24 is even
Enter a number(enter 0 to stop)
23
23 is odd
Enter a number(enter 0 to stop)
0
[root@localhost ~]#
```

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.

Code:

```
count_lines()
{
filename=$1
if [ -f "$filename" ]; then
	lines=$(wc -l < "$filename")
	echo "The file '$filename' has $lines lines."
else
	echo "Error: file '$filename' not found"
fi
}

count_lines "$1"
```

Output:

```
[root@localhost ~]# bash count.sh hello.txt
The file 'hello.txt' has 12 lines.
```

Assignment 4:

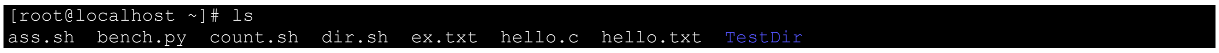
Write script that creates a directory named TestDir and inside it, creates ten files named Filel. txt, File2. txt, ... File10. txt. Each file should contain its filename as its content (e.g., Filel. txt contains "Filel.txt").

Code:

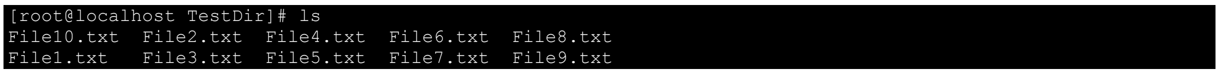
```
create_files()
{
dir=$1
if [ ! -d "$dir" ]; then
    mkdir "$dir"
fi
for ((i=1; i<=10; i++)); do
    filename="File$i.txt"
    echo "$filename" > "$dir/$filename"
done
}
create_files "TestDir"
```

Output:

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



```
[root@localhost ~]# ls
ass.sh  bench.py  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.

Code:

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

Output:

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

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.

Code:

```
#!/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.

Code:

```
if [ $# -ne 3 ]; then
    echo "Usage: $0 input_file old_file new_file"
fi
input=$1
old_text=$2
new_file=$3
output="${input%.txt}_modified.txt"
sed "s/$old_text/$new_text/g" "$input" > "$output"
echo "Replace done. result stored to $output"
```

Output:

```
[root@localhost ~]# bash edit.sh input.txt prakash naidu
Replace done. result stored to input_modified.txt
```