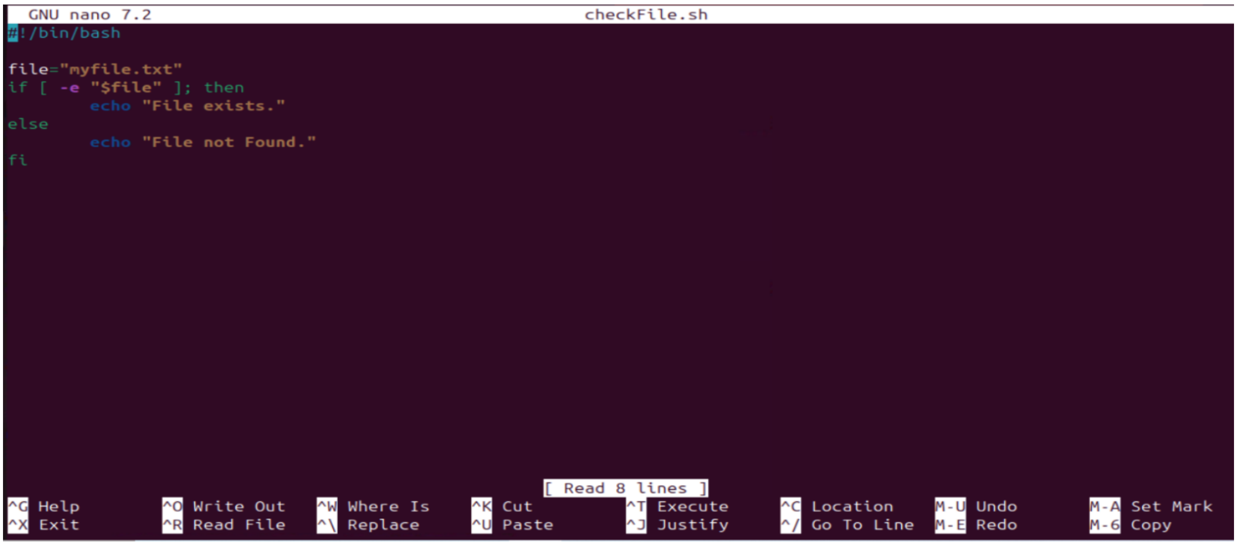
Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Shell script for checking a file in current working dir:

- ➔ Create a new file named checkFile.sh and provide it 'execute' permissions.

```
touch chechFile.sh

chmod 755 checkFile.sh
```

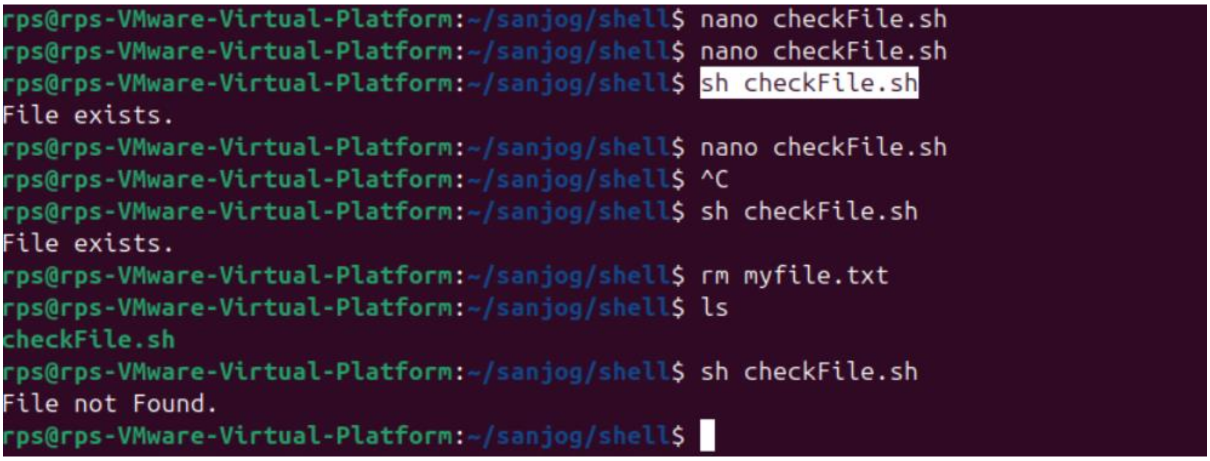
- ➔ using nano or other editor make changes to sh file and write code accordingly.



```
#!/bin/bash

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

```
[ Read 8 lines ]
```

- ➔ Execute command.

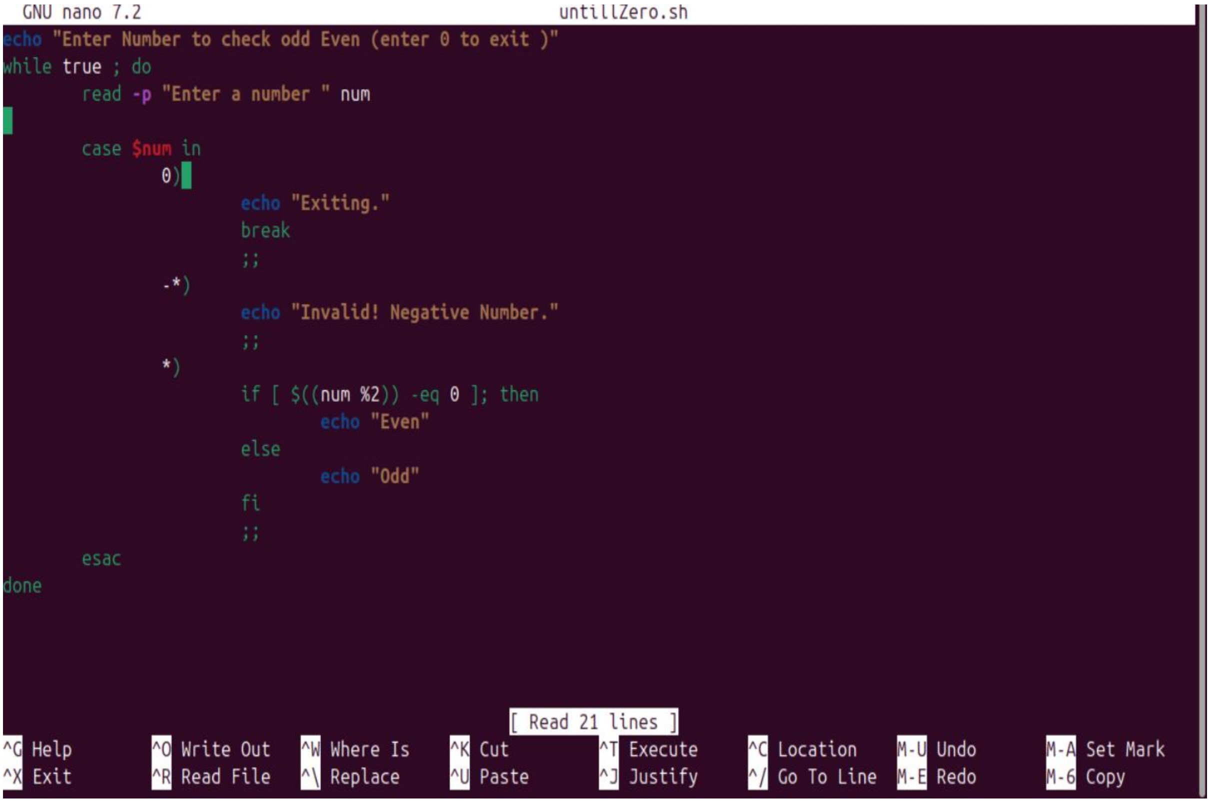


```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano checkFile.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano checkFile.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ sh checkFile.sh
File exists.
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano checkFile.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ ^C
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ sh checkFile.sh
File exists.
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ rm myfile.txt
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ ls
checkFile.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ sh checkFile.sh
File not Found.
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ 
```

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.

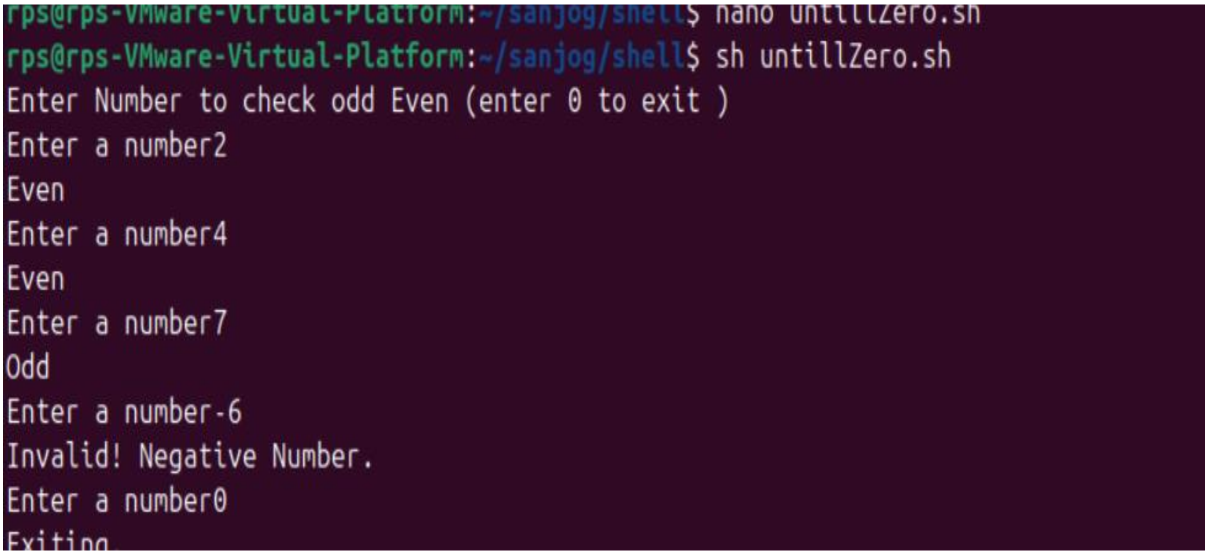
Shell script to read number till they write ‘0’ and check Odd & Even:

- ➔ The shell code is shown below.



```
echo "Enter Number to check odd Even (enter 0 to exit )"
while true ; do
        read -p "Enter a number " num

        case $num in
                0)
                        echo "Exiting."
                        break
                        ;;
                -*)
                        echo "Invalid! Negative Number."
                        ;;
                *)
                        if [ $((num %2)) -eq 0 ]; then
                                echo "Even"
                        else
                                echo "Odd"
                        fi
                        ;;
        esac
done
```

- ➔ Execution:

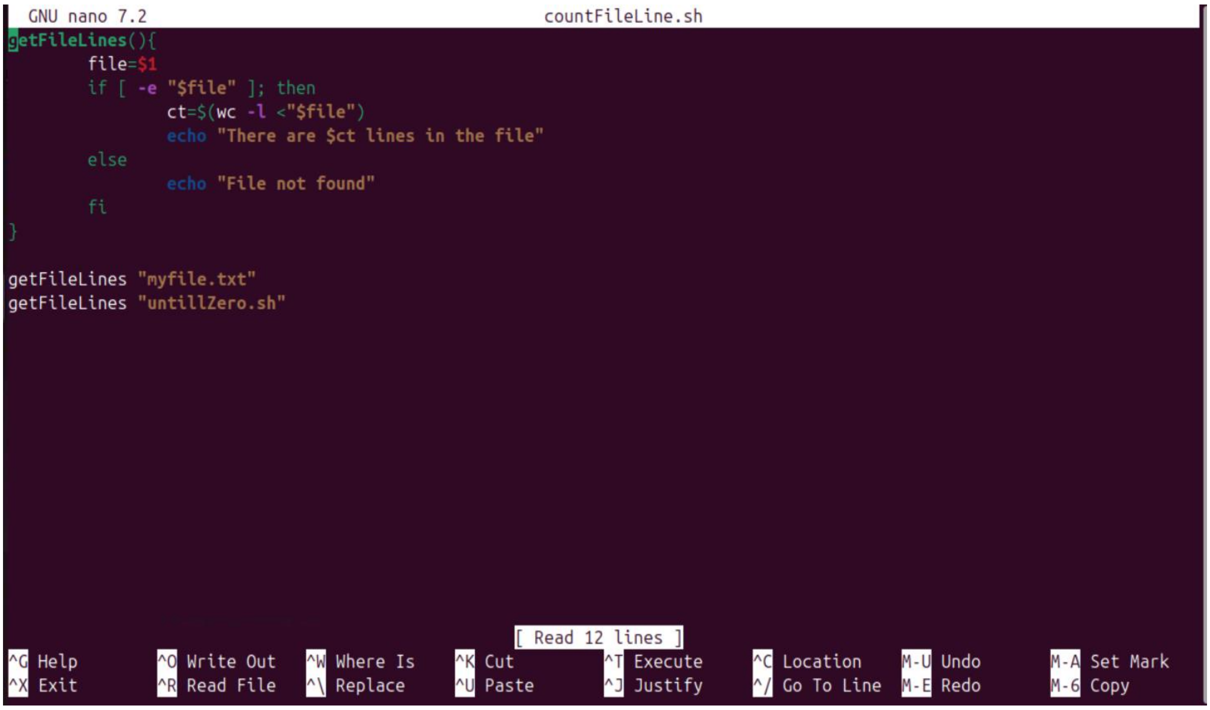


```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano untillZero.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ sh untillZero.sh
Enter Number to check odd Even (enter 0 to exit )
Enter a number2
Even
Enter a number4
Even
Enter a number7
Odd
Enter a number-6
Invalid! Negative Number.
Enter a number0
Exiting.
```

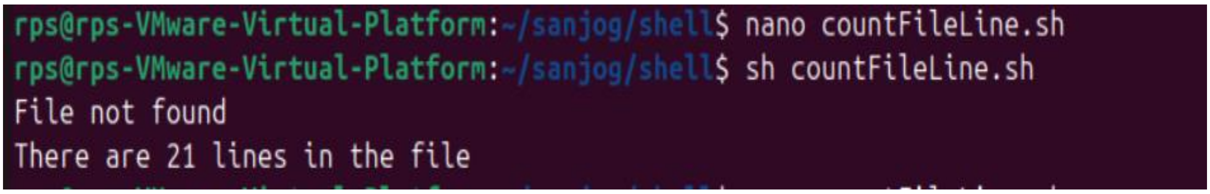
Assignment 3: Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.

Shell script to get number of lines in file using function:

➔ Code and execution shown below:



```
  GNU nano 7.2                                    countFileLine.sh
getFileLines(){
        file=$1
        if [ -e "$file" ]; then
                ct=$(wc -l <"$file")
                echo "There are $ct lines in the file"
        else
                echo "File not found"
        fi
}

getFileLines "myfile.txt"
getFileLines "untillZero.sh"

                                        [ Read 12 lines ]
```

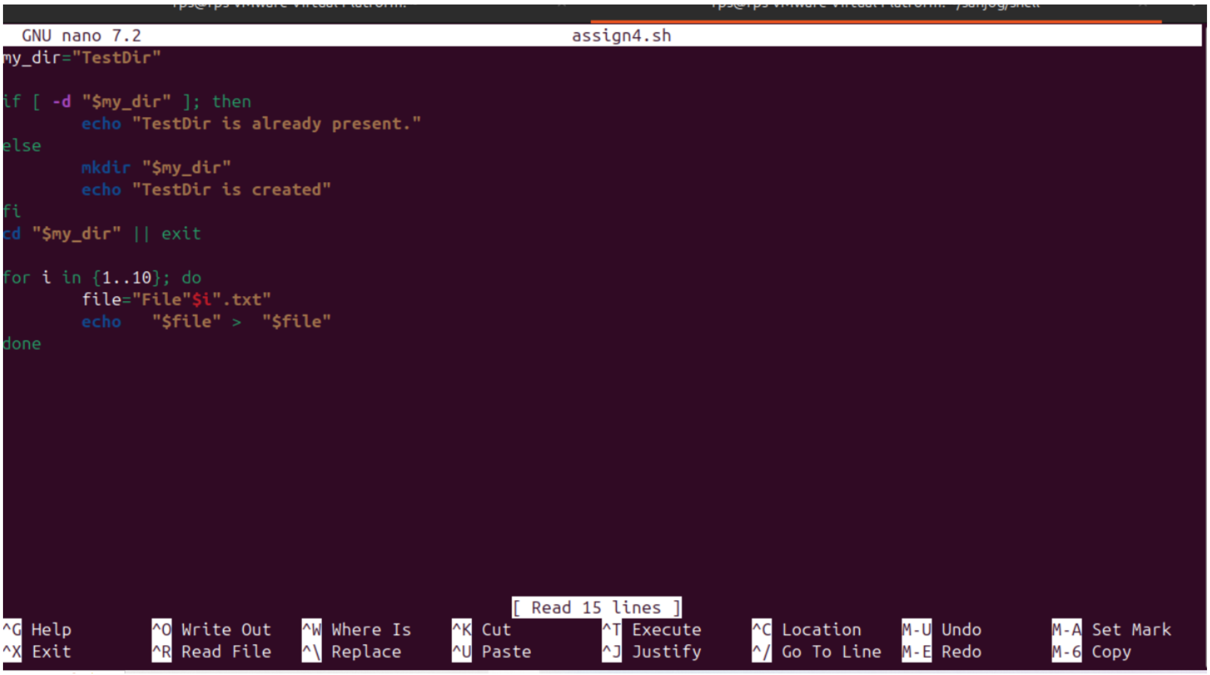


```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano countFileLine.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ sh countFileLine.sh
File not found
There are 21 lines in the file
```

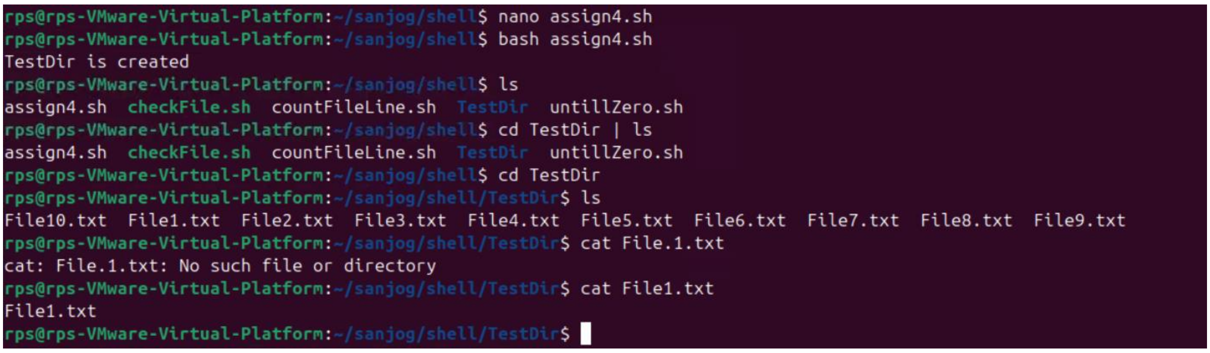
Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

Shell script to make files and its content:

- ➔ The code and its execution is shown below:



```
  GNU nano 7.2                                  assign4.sh
my_dir="TestDir"

if [ -d "$my_dir" ]; then
        echo "TestDir is already present."
else
        mkdir "$my_dir"
        echo "TestDir is created"
fi
cd "$my_dir" || exit

for i in {1..10}; do
        file="File"$i".txt"
        echo   "$file" >  "$file"
done

                                   [ Read 15 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



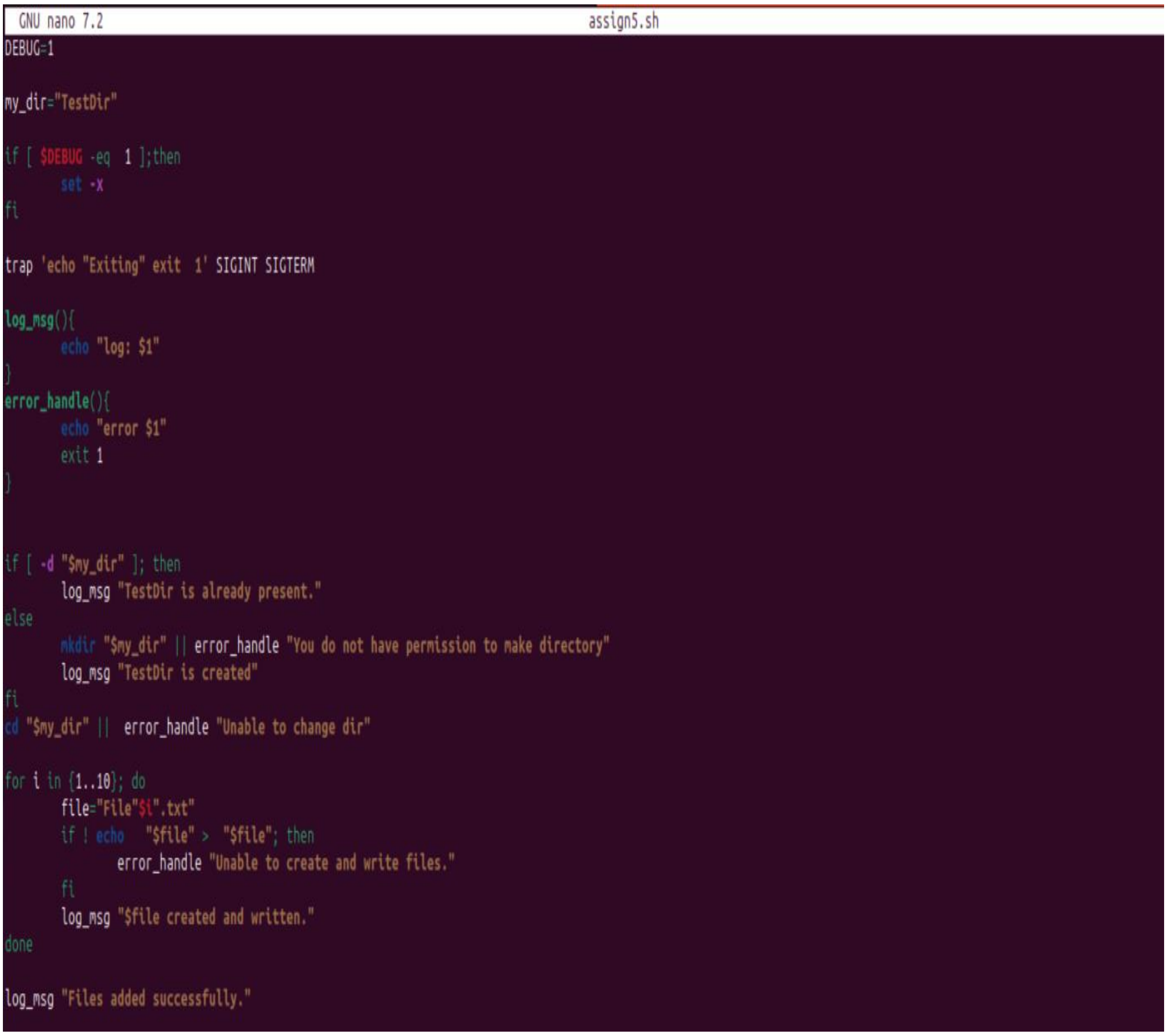
```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano assign4.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ bash assign4.sh
TestDir is created
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ ls
assign4.sh  checkFile.sh  countFileLine.sh  TestDir  untillZero.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ cd TestDir | ls
assign4.sh  checkFile.sh  countFileLine.sh  TestDir  untillZero.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ cd TestDir
rps@rps-VMware-Virtual-Platform:~/sanjog/shell/TestDir$ ls
File10.txt  File1.txt  File2.txt  File3.txt  File4.txt  File5.txt  File6.txt  File7.txt  File8.txt  File9.txt
rps@rps-VMware-Virtual-Platform:~/sanjog/shell/TestDir$ cat File.1.txt
cat: File.1.txt: No such file or directory
rps@rps-VMware-Virtual-Platform:~/sanjog/shell/TestDir$ cat File1.txt
File1.txt
rps@rps-VMware-Virtual-Platform:~/sanjog/shell/TestDir$ 
```

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

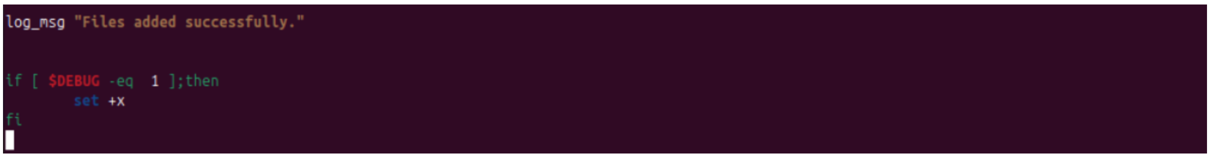
Add a debugging mode that prints additional information when enabled.

Shell script to for logging and error handling:

➔ The code and its execution is shown below:

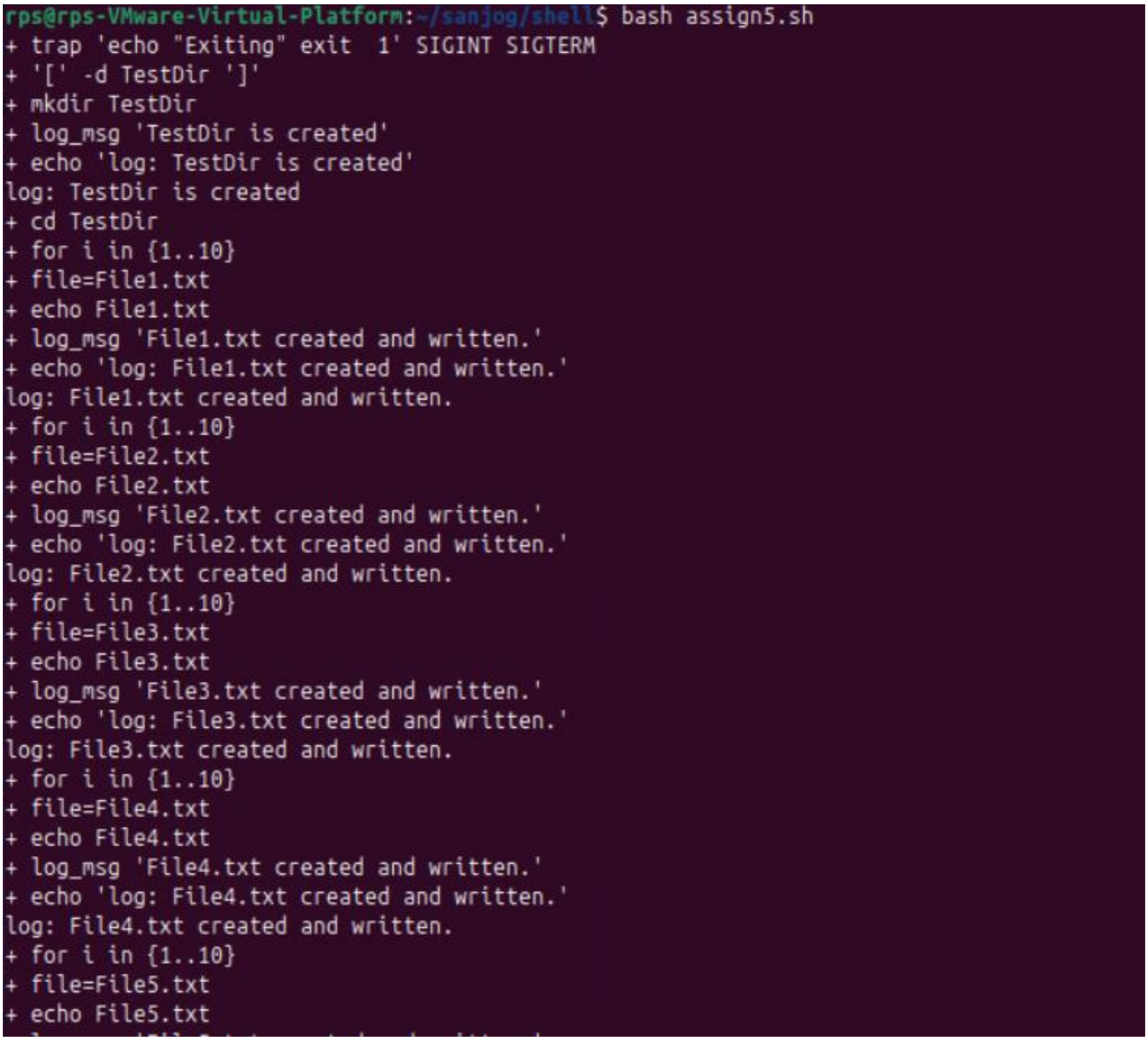


```
DEBUG=1

my_dir="TestDir"

if [ $DEBUG -eq  1 ];then
        set -x
fi

trap 'echo "Exiting" exit  1' SIGINT SIGTERM

log_msg(){
        echo "log: $1"
}
error_handle(){
        echo "error $1"
        exit 1
}


if [ -d "$my_dir" ]; then
        log_msg "TestDir is already present."
else
        mkdir "$my_dir" || error_handle "You do not have permission to make directory"
        log_msg "TestDir is created"
fi
cd "$my_dir" ||  error_handle "Unable to change dir"

for i in {1..10}; do
        file="File"$i".txt"
        if ! echo   "$file" >  "$file"; then
                error_handle "Unable to create and write files."
        fi
        log_msg "$file created and written."
done

log_msg "Files added successfully."
```



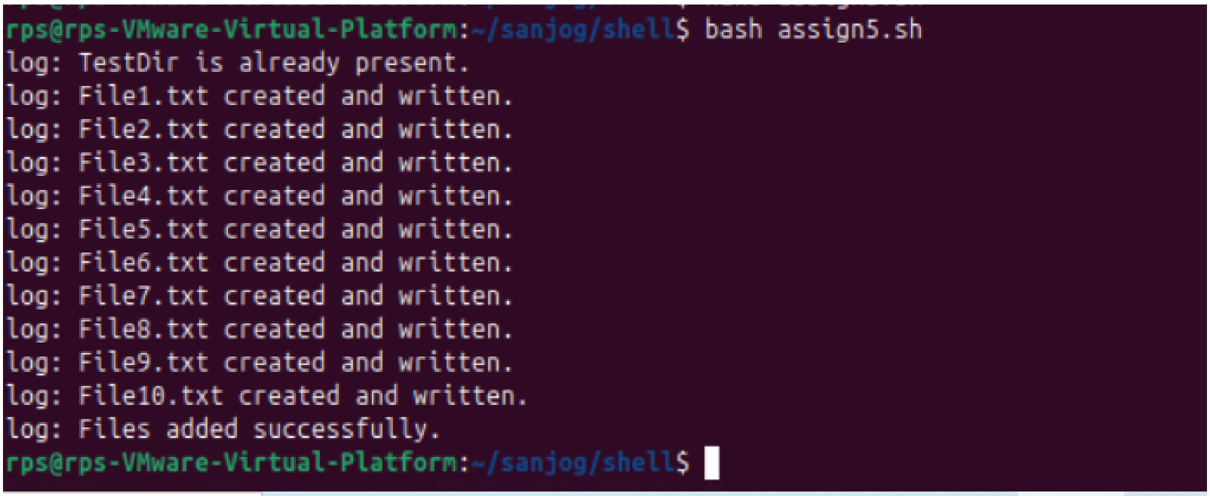
```
log_msg "Files added successfully."

if [ $DEBUG -eq  1 ];then
        set +x
fi
```

Execute using Debug=1:



```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ bash assign5.sh
+ trap 'echo "Exiting" exit  1' SIGINT SIGTERM
+ '[' -d TestDir ']'
+ mkdir TestDir
+ log_msg 'TestDir is created'
+ echo 'log: TestDir is created'
log: TestDir is created
+ cd TestDir
+ for i in {1..10}
+ file=File1.txt
+ echo File1.txt
+ log_msg 'File1.txt created and written.'
+ echo 'log: File1.txt created and written.'
log: File1.txt created and written.
+ for i in {1..10}
+ file=File2.txt
+ echo File2.txt
+ log_msg 'File2.txt created and written.'
+ echo 'log: File2.txt created and written.'
log: File2.txt created and written.
+ for i in {1..10}
+ file=File3.txt
+ echo File3.txt
+ log_msg 'File3.txt created and written.'
+ echo 'log: File3.txt created and written.'
log: File3.txt created and written.
+ for i in {1..10}
+ file=File4.txt
+ echo File4.txt
+ log_msg 'File4.txt created and written.'
+ echo 'log: File4.txt created and written.'
log: File4.txt created and written.
+ for i in {1..10}
+ file=File5.txt
+ echo File5.txt
```

Execute using Debug=0:



```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ bash assign5.sh
log: TestDir is already present.
log: File1.txt created and written.
log: File2.txt created and written.
log: File3.txt created and written.
log: File4.txt created and written.
log: File5.txt created and written.
log: File6.txt created and written.
log: File7.txt created and written.
log: File8.txt created and written.
log: File9.txt created and written.
log: File10.txt created and written.
log: Files added successfully.
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$
```

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.

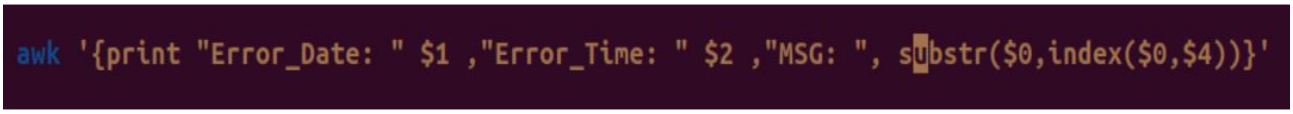
Data Processing with sed

Extracting Error and using grep and awk command:

- ➔ <u>grep:</u> it searches file for given cmd
- ➔ Now to search "Error" msg on we use condition as follows

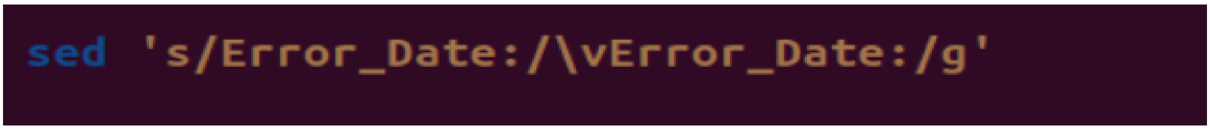
```
grep pattern filename

ex. grep "ERROR" sample_logs.log
```

- ➔ **|(pipe):** this will return output of previous cmd (grep cmd here) and works as input to RHS.
- ➔ **<u>awk:</u>** this command will make structure of given input by mainly using it part or columns.
- ➔ $1: Refers to the first field or column (i.e., Date for my file) in a line of input from grep.
- ➔ $2: Refers to the second field which is Time here
- ➔ **index ($0, $4):** This will get us this index of word after $3 which is "ERROR" or "INFO" word column
- ➔ **substr ($0, index ($0, $4)):** it returns sub string of each line from index from 4th column to end of line.



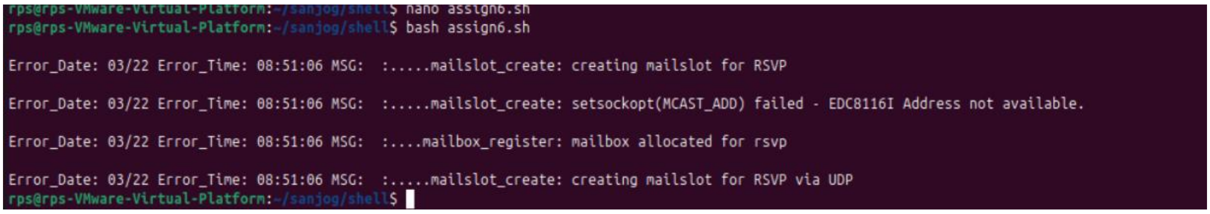
```
awk '{print "Error_Date: " $1 ,"Error_Time: " $2 ,"MSG: ", substr($0,index($0,$4))}'
```

- ➔ After getting the required data we can now process to show better to user using sed
- ➔ sed: it is a stream editor used for text transformations.



```
sed 's/Error_Date:/\vError_Date:/g'
```

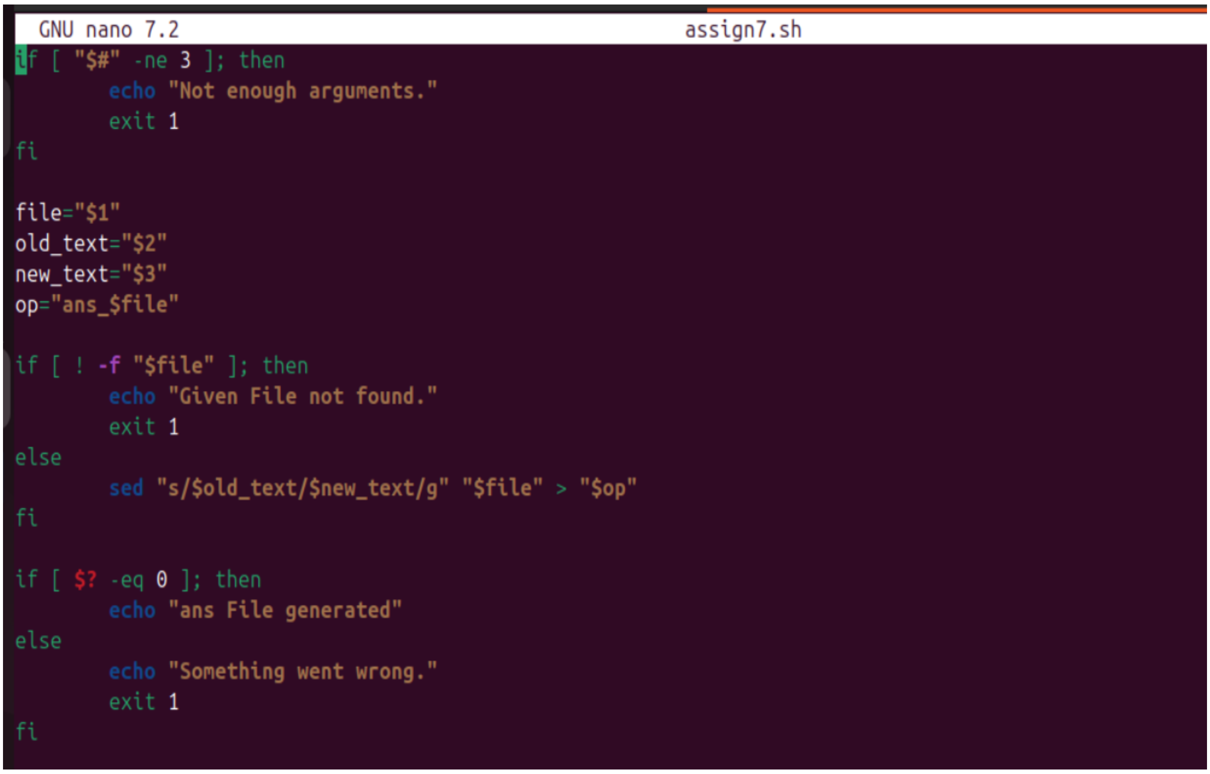
- ➔ s: Stands for substitution.
- ➔ Error_Date: Target text to search.
- ➔ \v: vertical tab.
- ➔ \vError_Date: add a vertical tab before Error Date text.
- ➔ g: Applies the substitution globally to all matches in the input like regular regex.



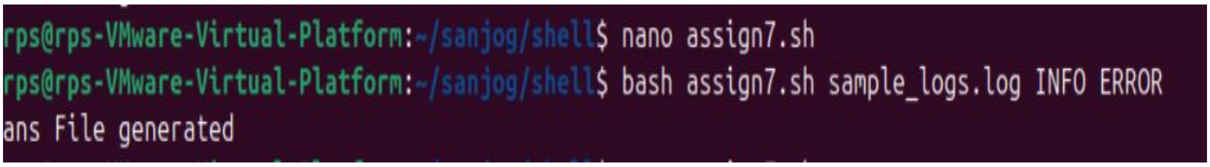
```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano assign6.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ bash assign6.sh

Error_Date: 03/22 Error_Time: 08:51:06 MSG:  :.....mailslot_create: creating mailslot for RSVP

Error_Date: 03/22 Error_Time: 08:51:06 MSG:  :.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.

Error_Date: 03/22 Error_Time: 08:51:06 MSG:  :....mailbox_register: mailbox allocated for rsvp

Error_Date: 03/22 Error_Time: 08:51:06 MSG:  :.....mailslot_create: creating mailslot for RSVP via UDP
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ 
```

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.

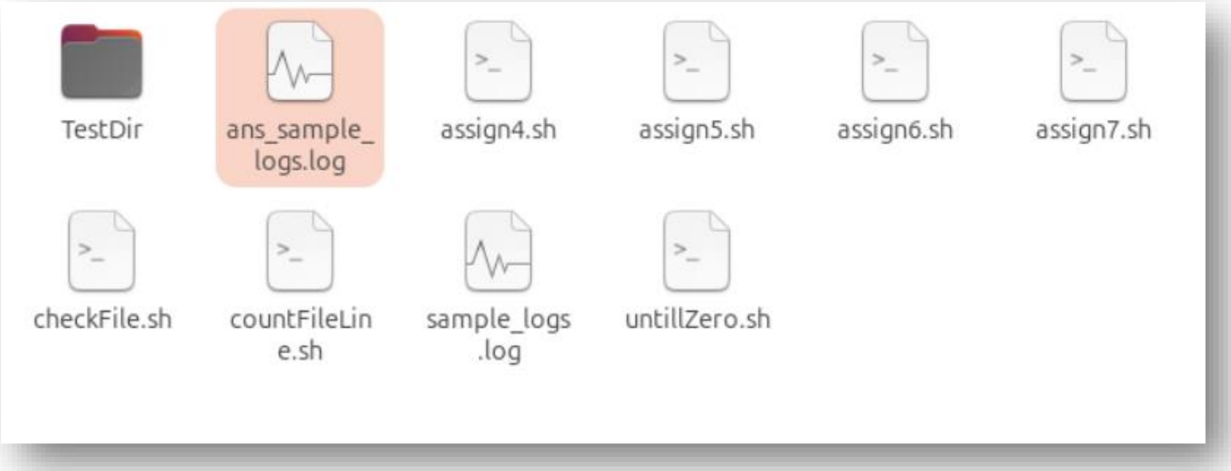
Replace words from one file and generate Ans:



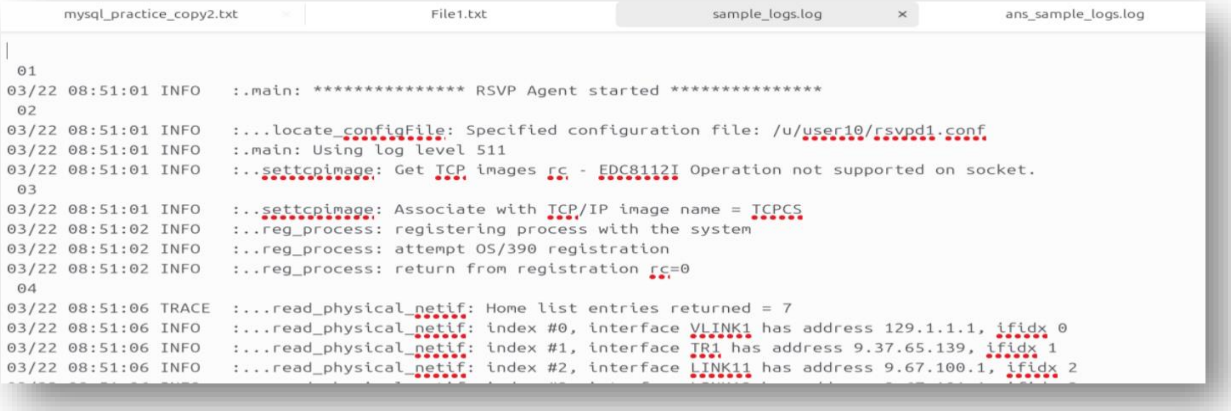
```
GNU nano 7.2                                    assign7.sh
if [ "$#" -ne 3 ]; then
        echo "Not enough arguments."
        exit 1
fi

file="$1"
old_text="$2"
new_text="$3"
op="ans_$file"

if [ ! -f "$file" ]; then
        echo "Given File not found."
        exit 1
else
        sed "s/$old_text/$new_text/g" "$file" > "$op"
fi

if [ $? -eq 0 ]; then
        echo "ans File generated"
else
        echo "Something went wrong."
        exit 1
fi
```



```
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ nano assign7.sh
rps@rps-VMware-Virtual-Platform:~/sanjog/shell$ bash assign7.sh sample_logs.log INFO ERROR
ans File generated
```

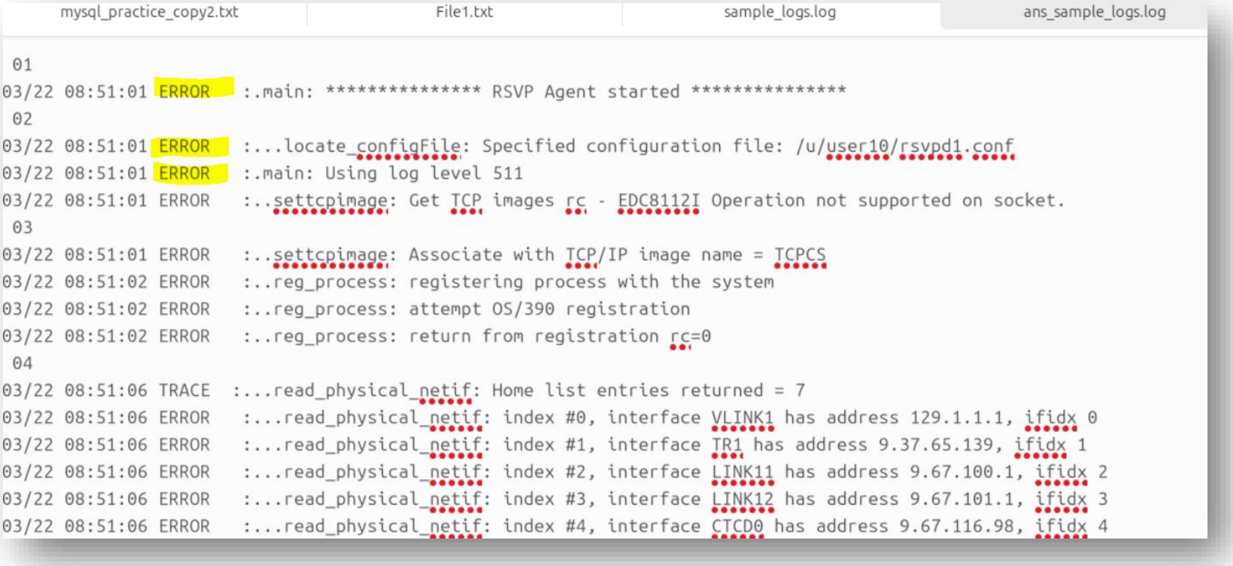


➔ Sample_log file



```
 01
03/22 08:51:01 INFO   :.main: *************** RSVP Agent started ***************
 02
03/22 08:51:01 INFO   :...locate_configFile: Specified configuration file: /u/user10/rsvpd1.conf
03/22 08:51:01 INFO   :.main: Using log level 511
03/22 08:51:01 INFO   :..settcpimage: Get TCP images rc - EDC8112I Operation not supported on socket.
 03
03/22 08:51:01 INFO   :..settcpimage: Associate with TCP/IP image name = TCPCS
03/22 08:51:02 INFO   :..reg_process: registering process with the system
03/22 08:51:02 INFO   :..reg_process: attempt OS/390 registration
03/22 08:51:02 INFO   :..reg_process: return from registration rc=0
 04
03/22 08:51:06 TRACE  :...read_physical_netif: Home list entries returned = 7
03/22 08:51:06 INFO   :...read_physical_netif: index #0, interface VLINK1 has address 129.1.1.1, ifidx 0
03/22 08:51:06 INFO   :...read_physical_netif: index #1, interface TR1 has address 9.37.65.139, ifidx 1
03/22 08:51:06 INFO   :...read_physical_netif: index #2, interface LINK11 has address 9.67.100.1, ifidx 2
```

➔ Output File ans_sample_logs



```
 01
03/22 08:51:01 ERROR   :.main: *************** RSVP Agent started ***************
 02
03/22 08:51:01 ERROR   :...locate_configFile: Specified configuration file: /u/user10/rsvpd1.conf
03/22 08:51:01 ERROR   :.main: Using log level 511
03/22 08:51:01 ERROR   :..settcpimage: Get TCP images rc - EDC8112I Operation not supported on socket.
 03
03/22 08:51:01 ERROR   :..settcpimage: Associate with TCP/IP image name = TCPCS
03/22 08:51:02 ERROR   :..reg_process: registering process with the system
03/22 08:51:02 ERROR   :..reg_process: attempt OS/390 registration
03/22 08:51:02 ERROR   :..reg_process: return from registration rc=0
 04
03/22 08:51:06 TRACE  :...read_physical_netif: Home list entries returned = 7
03/22 08:51:06 ERROR   :...read_physical_netif: index #0, interface VLINK1 has address 129.1.1.1, ifidx 0
03/22 08:51:06 ERROR   :...read_physical_netif: index #1, interface TR1 has address 9.37.65.139, ifidx 1
03/22 08:51:06 ERROR   :...read_physical_netif: index #2, interface LINK11 has address 9.67.100.1, ifidx 2
03/22 08:51:06 ERROR   :...read_physical_netif: index #3, interface LINK12 has address 9.67.101.1, ifidx 3
03/22 08:51:06 ERROR   :...read_physical_netif: index #4, interface CTCD0 has address 9.67.116.98, ifidx 4
```

Important Note:

- **Ref: https://www.ibm.com/docs/en/zos/2.4.0?topic=problems-example-log-file for Sample_log file.**
- **<u>Snipping tool</u> for cropping, copying and taking ss of outputs and codes**