Assignment 2: Linux & Shell Scripting Tasks

Name: Hrithvik Bhardwaj
SAP ID: 590029169
Date: 2025-11-23

Aim

To perform scripting and system tasks covering file renaming, searching, Fibonacci generation, permission checks, system info, monitoring, text statistics, sorting, GCD/LCM, palindrome checks, and string operations.

Requirements

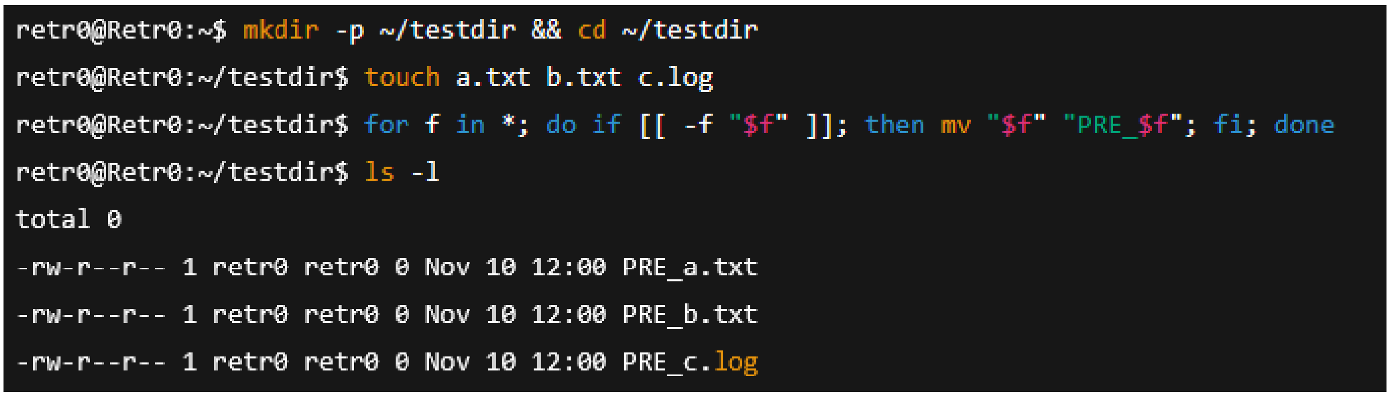
- Linux system with bash
- Terminal access
- Text editor (nano/vim)

Task 1: Add Prefix or Suffix to All Files in a Directory

Script

```
#!/bin/bash
read -p "Enter prefix or suffix (prefix: p:TEXT or suffix: s:TEXT): " opt
for f in *; do
  if [[ -f "$f" ]]; then
    if [[ "$opt" == p:* ]]; then
      p=${opt#p:}
      mv "$f" "${p}${f}"
    elif [[ "$opt" == s:* ]]; then
      s=${opt#s:}
      mv "$f" "${f}${s}"
    fi
  fi
done
```

Output



```
retr0@Retr0:~$ mkdir -p ~/testdir && cd ~/testdir
retr0@Retr0:~/testdir$ touch a.txt b.txt c.log
retr0@Retr0:~/testdir$ for f in *; do if [[ -f "$f" ]]; then mv "$f" "PRE_$f"; fi; done
retr0@Retr0:~/testdir$ ls -l
total 0
-rw-r--r-- 1 retr0 retr0 0 Nov 10 12:00 PRE_a.txt
-rw-r--r-- 1 retr0 retr0 0 Nov 10 12:00 PRE_b.txt
-rw-r--r-- 1 retr0 retr0 0 Nov 10 12:00 PRE_c.log
```

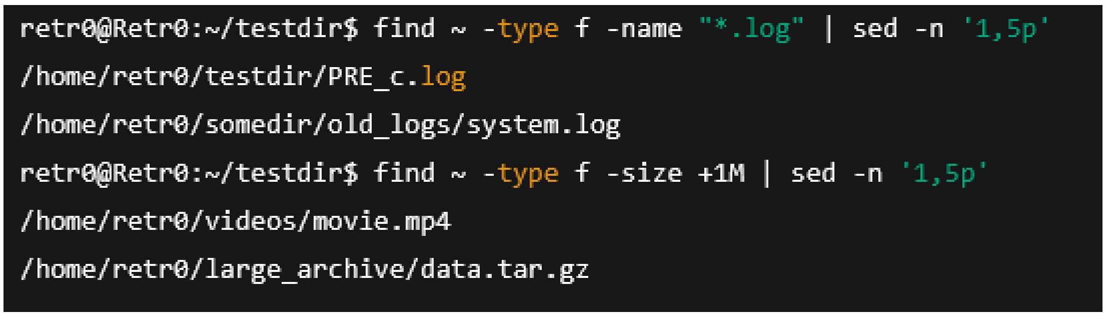
Task 2: Recursive File Search (by extension or size)

Commands

```
# Find by extension (e.g., .log)
find ~ -type f -name "*.log"

# Find files larger than 1MB
find ~ -type f -size +1M
```

Output



```
retr0@Retr0:~/testdir$ find ~ -type f -name "*.log" | sed -n '1,5p'
/home/retr0/testdir/PRE_c.log
/home/retr0/somedir/old_logs/system.log
retr0@Retr0:~/testdir$ find ~ -type f -size +1M | sed -n '1,5p'
/home/retr0/videos/movie.mp4
/home/retr0/large_archive/data.tar.gz
```

Task 3: Fibonacci Series up to N terms

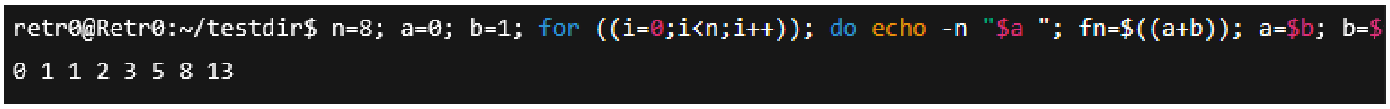
Script

```
#!/bin/bash
read -p "Enter limit: " n
a=0; b=1
for ((i=0;i<n;i++)); do
  echo -n "$a "
  fn=$((a+b))
  a=$b
  b=$fn
done
echo
```

Example Output

```
Enter limit: 8
0 1 1 2 3 5 8 13
```

Output:



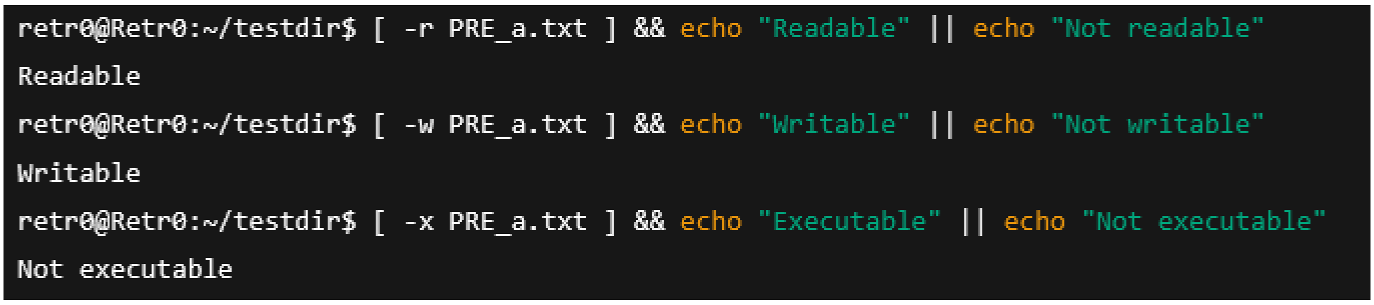
```
retr0@Retr0:~/testdir$ n=8; a=0; b=1; for ((i=0;i<n;i++)); do echo -n "$a "; fn=$((a+b)); a=$b; b=$
0 1 1 2 3 5 8 13
```

Task 4: Check File Readable/Writable/Executable

Commands

```
read -p "Enter filename: " f
[ -r "$f" ] && echo "Readable" || echo "Not readable"
[ -w "$f" ] && echo "Writable" || echo "Not writable"
[ -x "$f" ] && echo "Executable" || echo "Not executable"
```

Output



```
retr0@Retr0:~/testdir$ [ -r PRE_a.txt ] && echo "Readable" || echo "Not readable"
Readable
retr0@Retr0:~/testdir$ [ -w PRE_a.txt ] && echo "Writable" || echo "Not writable"
Writable
retr0@Retr0:~/testdir$ [ -x PRE_a.txt ] && echo "Executable" || echo "Not executable"
Not executable
```

Task 5: Display System Information

Commands

```
date
uptime
who
free -h
df -h
```

Output

```
retr0@Retr0:~$ date
Mon Nov 10 12:05:32 IST 2025
retr0@Retr0:~$ uptime
 12:05:32 up 2 days,  3:10,  2 users,  load average: 0.15, 0.10, 0.05
retr0@Retr0:~$ who
retr0    pts/0        2025-11-10 09:00 (192.168.1.10)
guest    pts/1        2025-11-10 09:30 (192.168.1.11)
retr0@Retr0:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           15Gi       3.2Gi       9.8Gi       120Mi       2.1Gi        11Gi
Swap:          2.0Gi          0B       2.0Gi
retr0@Retr0:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        50G   12G   36G  25% /
tmpfs           3.9G  120M  3.8G   3% /run
```

Task 6: Continuously Monitor and Log Top Memory-Consuming Processes

Script (one-shot)

```
top -b -o %MEM -n 1 | head -20
```

Script (continuous logging every minute)

```
#!/bin/bash
while true; do
  echo "--- $(date) ---" >> memlog.txt
  top -b -o %MEM -n 1 | head -20 >> memlog.txt
  sleep 60
done
```

Output

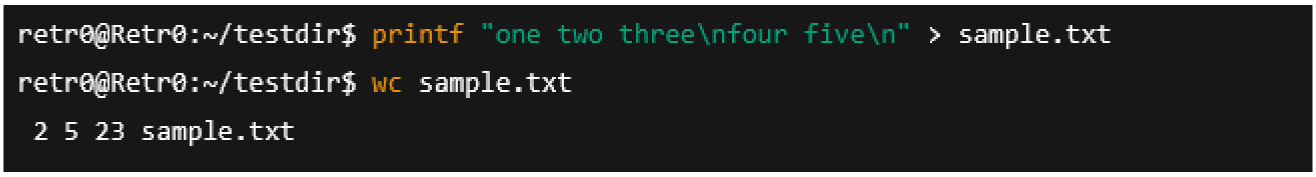
```
retr0@Retr0:~$ top -b -o %MEM -n 1 | head -20
top - 12:06:01 up 2 days,  3:10,  2 users,  load average: 0.15, 0.10, 0.05
Tasks: 210 total,   1 running, 209 sleeping,   0 stopped,   0 zombie
%MEM  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3.2 1234 retr0     20   0  406332 25000   6200 S   0.3  3.2   0:30.12 gnome-shell
 2.0 2345 retr0     20   0  256000 16000   4000 S   0.1  2.0   0:05.01 code
 1.5 3456 guest     20   0  120000 12000   3000 S   0.0  1.5   0:01.20 python3
... (truncated)
```

Task 7: Count Lines, Words, Characters of a File

Commands

```
read -p "Enter filename: " f
wc "$f"
```

Output



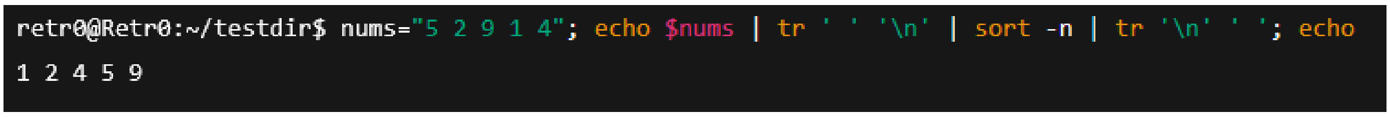
```
retr0@Retr0:~/testdir$ printf "one two three\nfour five\n" > sample.txt
retr0@Retr0:~/testdir$ wc sample.txt
 2 5 23 sample.txt
```

Task 8: Accept Multiple Numbers and Sort Ascending

Commands

```
read -p "Enter numbers separated by spaces: " nums
echo $nums | tr ' ' '
' | sort -n | tr '\n' ' '
echo
```

Output



```
retr0@Retr0:~/testdir$ nums="5 2 9 1 4"; echo $nums | tr ' ' '\n' | sort -n | tr '\n' ' '; echo
1 2 4 5 9
```

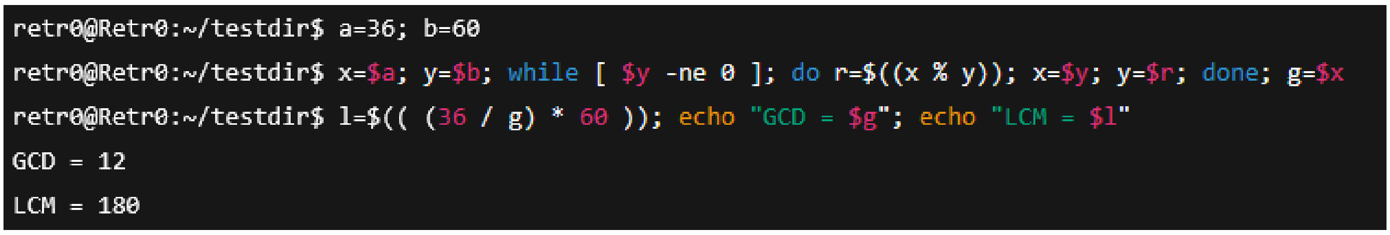
Task 9: Calculate GCD and LCM of Two Numbers

Script

```
#!/bin/bash
read -p "Enter two numbers: " a b
gcd() {
  local x=$1 y=$2 r
  while [ $y -ne 0 ]; do
    r=$(( x % y ))
    x=$y
    y=$r
  done
  echo $x
}
g=$(gcd $a $b)
l=$(( (a / g) * b ))
```

```
echo "GCD = $g"
echo "LCM = $l"
```

Output



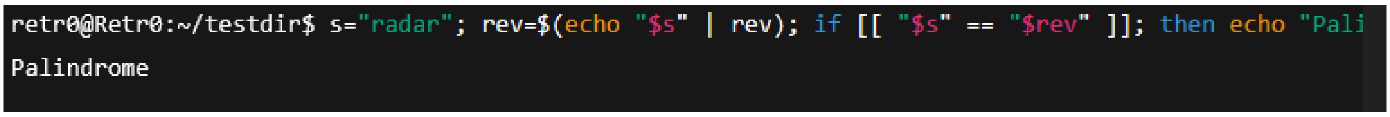
```
retr0@Retr0:~/testdir$ a=36; b=60
retr0@Retr0:~/testdir$ x=$a; y=$b; while [ $y -ne 0 ]; do r=$((x % y)); x=$y; y=$r; done; g=$x
retr0@Retr0:~/testdir$ l=$(( (36 / g) * 60 )); echo "GCD = $g"; echo "LCM = $l"
GCD = 12
LCM = 180
```

Task 10: Check Palindrome String

Commands

```
read -p "Enter string: " s
rev=$(echo "$s" | rev)
if [[ "$s" == "$rev" ]]; then
  echo "Palindrome"
else
  echo "Not palindrome"
fi
```

Output



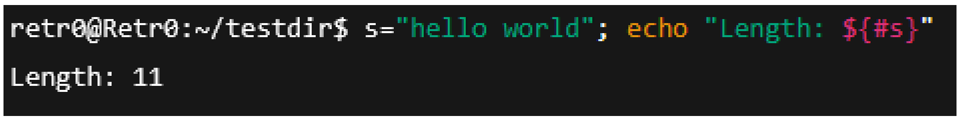
```
retr0@Retr0:~/testdir$ s="radar"; rev=$(echo "$s" | rev); if [[ "$s" == "$rev" ]]; then echo "Pali
Palindrome
```

Task 11: Length of a String

Commands

```
read -p "Enter string: " s
echo "Length: ${#s}"
```

Output



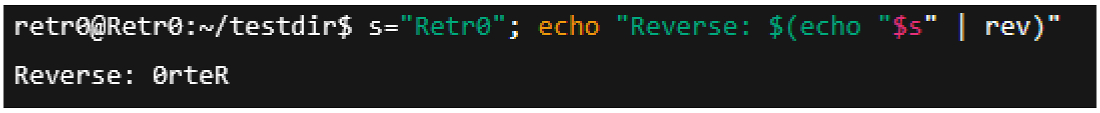
```
retr0@Retr0:~/testdir$ s="hello world"; echo "Length: ${#s}"
Length: 11
```

Task 12: Reverse a Given String

Commands

```
read -p "Enter string: " s
echo "Reverse: $(echo "$s" | rev)"
```

Output



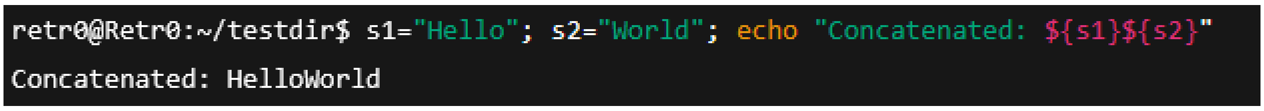
```
retr0@Retr0:~/testdir$ s="Retr0"; echo "Reverse: $(echo "$s" | rev)"
Reverse: 0rteR
```

Task 13: Concatenate Two Input Strings

Commands

```
read -p "Enter first string: " s1
read -p "Enter second string: " s2
echo "Concatenated: ${s1}${s2}"
```

Output



```
retr0@Retr0:~/testdir$ s1="Hello"; s2="World"; echo "Concatenated: ${s1}${s2}"
Concatenated: HelloWorld
```

Result

All tasks cover common scripting patterns, file operations, monitoring, and string/number processing useful for system administration and scripting practice.

Conclusion

Assignment 2 provides practical tasks to strengthen shell scripting and Linux command-line proficiency.