

ASSIGNMENT - 2

Activity 1: Write a script that adds a user-defined prefix or suffix to all files in a directory.

Screenshot :

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
```

Script Explanation:

- The script starts by taking user input for a directory path, a prefix, and a suffix.

- The `cd` command is used to change the working directory to the directory entered by the user.
- A for loop is used to iterate through all items in the directory.
- `[-f "$file"]` checks whether the current item is a regular file; if not, it is skipped.
- Parameter expansion is used to extract: `name` --> filename without extension (`${file%.*}`) `ext` --> file extension (`${file##*.}`)
- A new filename is constructed by concatenating prefix + original name + suffix.
- The `mv` command is used to rename each file to its new name.
- If the file has an extension, it is preserved; if not, the file is renamed without adding a dot.
- After all files are processed, the script prints "Renaming complete!" to indicate successful execution.

Terminal >

```
nongshim@ubuntu:~$ vim prefix.sh
nongshim@ubuntu:~$ bash prefix.sh
Enter prefix or suffix (prefix: p:TEXT or suffix: s:TEXT): p :
nongshim@ubuntu:~$ ls
Desktop  Downloads  lab2      Music     myfile.txt  MyLinuxFiles.tar.gz  Pictures  prefix.sh  readme.txt  Templates
Documents  file.txt   log.txt   myfile2.txt  MyLinuxFiles  notes.txt            practice_linux  Public    snap        Videos
nongshim@ubuntu:~$
```

Activity 2: Search recursively for files with a given extension or larger than a specified size.

Commands Used:

- **find** - searches for files inside the given directory and its subfolders.
- **~/Downloads/C_programming/lab1** - is the path where the search starts.
- **-type f -name "*.c"** - means show only files whose name ends with extension `.c`
- **find** - searches inside the given directory and all of its subfolders.
- **~/Downloads** - is the folder where the search begins.
- **-size +100k** - means show files larger than 100 kilobytes.

Screenshot :

Terminal >

```
nongshim@ubuntu:~$ find ~ -type f -name "*.log"
/home/nongshim/.local/share/gvfs-metadata/root-2cd760e6.log
/home/nongshim/.local/share/gvfs-metadata/home-cdb38925.log
/home/nongshim/Downloads/lab8/EXP8.log
/home/nongshim/Downloads/error.log
/home/nongshim/snap/firefox/common/.mozilla/firefox/Crash Reports/crash_helper_server.log
nongshim@ubuntu:~$ find ~ -type f -size +1M
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Documents.db-wal
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Video.db
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23FileSystem.db-wal
/home/nongshim/.cache/tracker3/files/meta.db
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23FileSystem.db
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Audio.db
/home/nongshim/.cache/tracker3/files/http%3A%2F%2Ftracker.api.gnome.org%2Fontology%2Fv3%2Ftracker%23Pictures.db-wal
```

Activity 3: *Generate Fibonacci series up to a given number of terms.*

Example Output:

Enter limit: 8

0 1 1 2 3 5 8 13

Screenshot :

Script >

```
#!/bin/bash

echo "Enter limit:"
read n

a=0
b=1

echo -n "$a $b "

for (( i=3; i<=n; i++ ))
do
    c=$((a + b))
    echo -n "$c "
    a=$b
    b=$c
done

echo
```

Script Explanation:

- The script **fibonacci.sh** takes a number **n** from the user to decide how many Fibonacci terms to print.
- It stores the first two Fibonacci numbers in variables **a = 0** and **b = 1**.
- It prints **0** and **1** before the loop begins.
- Inside the loop, each new number is calculated as **c = a + b**, printed, then the values of a and b are updated.
- The loop repeats until all Fibonacci numbers up to the entered limit are displayed.

Terminal >

```
nongshim@ubuntu:~$ vim filename.sh
nongshim@ubuntu:~$ source filename.sh
Enter limit:
45
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346
4157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733
nongshim@ubuntu:~$
```

Activity 4: Check if a file is readable, writable, or executable by the user.

Commands Used:

- **ls -l filename** – used to display file(s) with detailed information including permissions, owner, size, etc.

Script:

```
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"
~
```

Terminal >

```
nongshim@ubuntu:~$ vim executable.sh
nongshim@ubuntu:~$ source executable.sh
Enter filename: myfile.txt
Readable
Writable
Executable
nongshim@ubuntu:~$
```

Activity 5: Display system information (date, uptime, users, memory, disk usage).

Commands Used:

- **date** – shows the current system date and time.
- **uptime** – displays how long the system has been running along with load average
- **who** – shows the list of users currently logged into the system.

- **free -h** – shows the total, used, and available memory in a human-readable format.
- **df -h** – displays disk usage of all mounted filesystems in a human-readable format.

Screenshot :

Terminal >

```
nongshim@ubuntu:~$ date
Sat Nov 29 11:24:11 AM IST 2025
nongshim@ubuntu:~$ uptime
 11:24:17 up 29 min,  1 user,  load average: 0.37, 0.43, 0.45
nongshim@ubuntu:~$ who
nongshim seat0          2025-11-29 10:56 (login screen)
nongshim tty2           2025-11-29 10:56 (tty2)
nongshim@ubuntu:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:            8.6Gi        2.1Gi        4.7Gi        130Mi        1.9Gi        6.4Gi
Swap:            0B           0B           0B
nongshim@ubuntu:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           879M  1.6M  878M   1% /run
/dev/sda2       30G   8.1G   20G  29% /
tmpfs           4.3G   0    4.3G   0% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           879M  132K  879M   1% /run/user/1000
/dev/sr0        51M   51M    0 100% /media/nongshim/VBox_GAs_7.2.4
nongshim@ubuntu:~$
```

Activity 6: *Continuously monitor and log top memory-consuming processes.*

Screenshot :

Script >

```
#!/bin/bash

logfile="memory_log.txt"

while true
do
    echo "----- $(date)-----" >> "$logfile"
    ps aux --sort=-%mem | head -n 6 >> "$logfile"
    echo "" >> "$logfile"
    sleep 5
done
~
```

Script Explanation:

- The script creates a log file named **memory_log.txt** to store the output of memory-consuming processes.
- A while true loop runs continuously so that monitoring does not stop until the user cancels it.
- Inside the loop, current date and time are written to the log file to mark when the reading was taken.
- The command **ps aux --sort=-%mem | head -n 6** is used to list the top 5 processes that are using the highest memory.
- The output is appended to the log file every time instead of replacing the previous data.
- The script waits for 5 seconds using sleep 5 before checking again.
- The loop keeps repeating, which means the script continuously monitors memory usage until the user stops it manually using Ctrl + C

Terminal >

Activity 7: Take a filename as input and display the number of lines, words, and characters.

```
#!/bin/bash

echo "Enter filename:"
read file

echo "Lines:      $(wc -l < "$file")"
echo "Words:      $(wc -w < "$file")"
echo "Characters: $(wc -m < "$file")"
```

- **echo "Enter filename:"** displays a message asking the user to type a filename.
- **read file** stores the filename entered by the user into the variable file.

- ***wc -l < "\$file"*** counts the number of lines in the file and prints it with the label ***"Lines:"***
- ***wc -w < "\$file"*** counts the number of words in the file and prints it with the label ***"Words:"***
- ***wc -m < "\$file"*** counts the number of characters in the file and prints it with the label ***"Characters:"***
- The use of ***< "\$file"*** ensures that only the numeric result is printed, not the filename.

Terminal >

```
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ vim counter.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ chmod 744 counter.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ source counter.sh
Enter filename:
file1.txt
Lines:      5
Words:      5
Characters: 34
nongshim@ubuntu:~/Downloads/MyLinuxFiles$
```

Activity 8: *Accept multiple numbers and sort them in ascending order.*

Screenshot :

Script >

```
#!/bin/bash

echo "Enter numbers:"
read nums

for n in $nums
do
    echo $n
done | sort -n

~
~
```

Script Explanation:

- The script asks the user to input multiple numbers and stores them in the variable **nums**
- The **for n in \$nums** loop takes each number from the input one by one. □ Each number is printed on a new line using **echo \$n**
- The output of the loop is passed to **sort -n**, which sorts the numbers in ascending (numeric) order.

Terminal >

```
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ vim sort.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ chmod 744 sort.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ source sort.sh
Enter numbers:
76 875 73 7 8
7
8
73
76
875
nongshim@ubuntu:~/Downloads/MyLinuxFiles$
```

Activity 9: Calculate the GCD and LCM of two given numbers.

Screenshot :

Script >

```
#!/bin/bash

echo "Enter first number:"
read a

echo "Enter second number:"
read b

x=$a
y=$b

# GCD calculation
while [ $a -ne $b ]
do
    if [ $a -gt $b ]
    then
        a=$((a - b))
    else
        b=$((b - a))
    fi
done

gcd=$a
lcm=$(( (x * y) / gcd ))

echo "GCD = $gcd"
echo "LCM = $lcm"
```

Script Explanation:

- The script takes two numbers from the user and stores copies of them in **x** and **y** so they can be used later for **LCM** calculation.
- A while loop runs until both numbers become equal by repeatedly subtracting the smaller number from the larger one.
- When both numbers become equal, that common value is the **GCD**.
- The **LCM** is calculated using the formula: **LCM = (x * y) / GCD**.
- Finally, the script displays the calculated **GCD** and **LCM** on the screen.

Terminal >

```
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ vim gcd_lcm.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ chmod 744 gcd_lcm.sh
nongshim@ubuntu:~/Downloads/MyLinuxFiles$ source gcd_lcm.sh
Enter first number:
56
Enter second number:
34
GCD = 2
LCM = 952
nongshim@ubuntu:~/Downloads/MyLinuxFiles$
```

Activity 10: Check whether an entered string is a palindrome or not.

Screenshot :

Script >

```
#!/bin/bash

echo "Enter a string:"
read str

rev=$(echo "$str" | rev)

if [ "$str" = "$rev" ]
then
    echo "It is a palindrome"
else
    echo "It is not a palindrome"
fi

~
```

Script Explanation:

- The script asks the user to enter a string and stores it in the variable **str**
- The **rev** command is used to reverse the string and store the reversed result in the variable **rev**
- The script then compares the original string (**str**) with the reversed string (**rev**).
- If both strings are exactly the same, the script prints "**It is a palindrome**".
- If the original and reversed strings are different, the script prints "**It is not a palindrome**".

Terminal >

```
nongshim@ubuntu:~/Downloads$ vim palindrome.sh
nongshim@ubuntu:~/Downloads$ chmod 744 palindrome.sh
nongshim@ubuntu:~/Downloads$ bash palindrome.sh
Enter a string:
civic
It is a palindrome
nongshim@ubuntu:~/Downloads$ bash palindrome.sh
Enter a string:
create
It is not a palindrome
nongshim@ubuntu:~/Downloads$
```

Activity 11: Calculate and display the length of a string.

Screenshot :

Script >

```
#!/bin/bash

echo -n "Enter a string: "
read str

len=${#str}
echo "Length of the string = $len"

~
```

Script Explanation:

- The script reads a string from the user and stores it in the variable **str**
- **\${#str}** is used to calculate the total number of characters in the string.
- The script then prints the length of the string on the screen.

Terminal >

```
nongshim@ubuntu:~/Downloads$ vim revstr.sh
nongshim@ubuntu:~/Downloads$ chmod 744 revstr.sh
nongshim@ubuntu:~/Downloads$ bash revstr.sh
Enter a string: scandrum
Length of the string = 8
nongshim@ubuntu:~/Downloads$
```

Activity 12: Reverse a given string.

Screenshot :

Script >

```
#!/bin/bash

echo "Enter a string:"
read str

rev=$(echo "$str" | rev)

echo "Reversed string: $rev"
~
~
~
```

Script Explanation:

- The script takes a string from the user and stores it in a variable **str** .
- The **rev** command is used to reverse the characters of the string.
- Finally, the reversed string is displayed on the screen using **echo** .

Terminal >

```
nongshim@ubuntu:~/Downloads$ vim strlen.sh
nongshim@ubuntu:~/Downloads$ chmod 744 strlen.sh
nongshim@ubuntu:~/Downloads$ bash strlen.sh
Enter a string:
I'm a programmer
Reversed string: remmargorp a m'I
nongshim@ubuntu:~/Downloads$
```

Activity 13: Concatenate two input strings.

Screenshot :

Script >

```
#!/bin/bash

echo "Enter first string:"
read str1

echo "Enter second string:"
read str2

result="$str1$str2"

echo "Concatenated string: $result"
~
~
```

Script Explanation:

- The script asks the user to enter two strings and stores them in variables **str1** and **str2**.
- It combines both strings by placing them next to each other inside a single variable: **result="\$str1\$str2"**.
- This operation is called **concatenation**, meaning joining two strings together without spaces unless added manually.
- Finally, the script prints the concatenated result on the screen using **echo**.

Terminal >

```
nongshim@ubuntu:~/Downloads$ vim cocatstr.sh
nongshim@ubuntu:~/Downloads$ chmod 744 cocatstr.sh
nongshim@ubuntu:~/Downloads$ bash cocatstr.sh
Enter first string:
heuristic programm
Enter second string:
clater things
Concatenated string: heuristic programmclater things
nongshim@ubuntu:~/Downloads$
```

