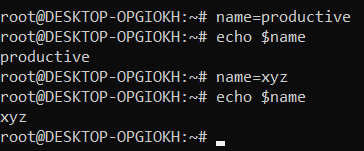
***CDAC MUMBAI***

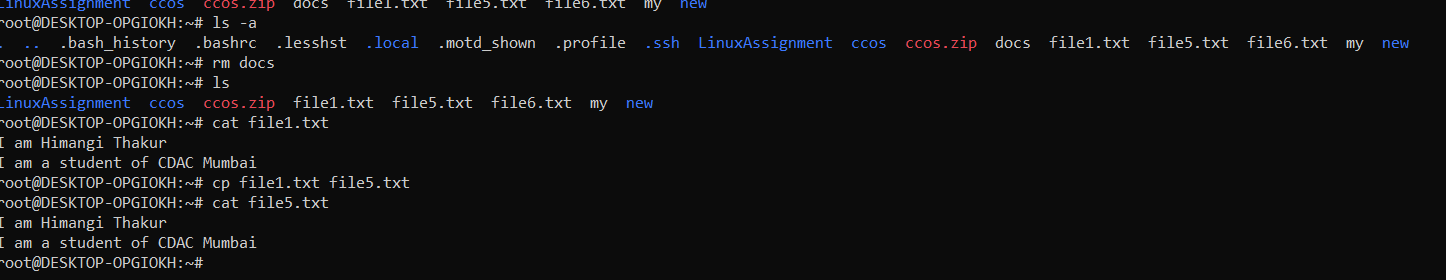
**Concepts of Operating System**

**Assignment 2**

***Part A***

**What will the following commands do?**

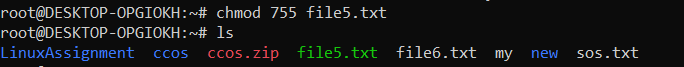
* **echo "Hello, World!"** – echo used for print the text to the terminal
* **name="Productive"** – name=any text, To store the text and we access this text using echo $name
* 
* **touch file.txt** - Useful for creating multiple empty files at once.
* **ls -a** - **Lists all files and directories,** including **hidden files**
* **rm file.txt** – to delete any file, remember this can’t be retrieve
* **cp file1.txt file2.txt** – copy text



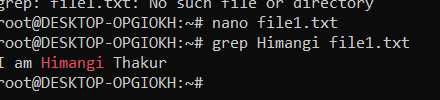
* **mv file.txt /path/to/directory/** - mv command is used for renaming

mv oldname.txt newname.txt

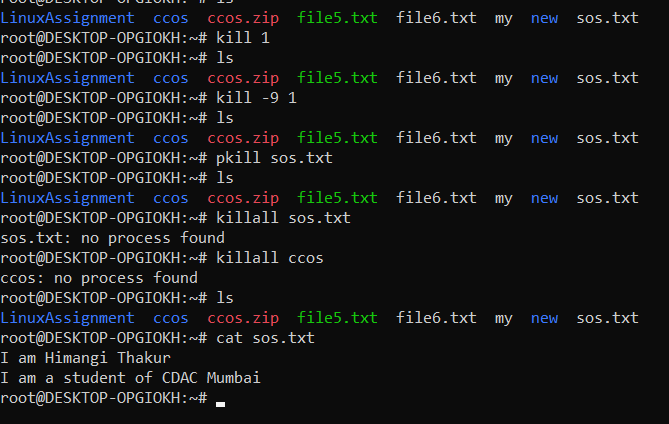
* **chmod 755 script.sh** – used for file permission read,write and execute permission for owner and read execute permission for other.



* **grep "pattern" file.txt** – grep used for highlighting the text



* kill PID

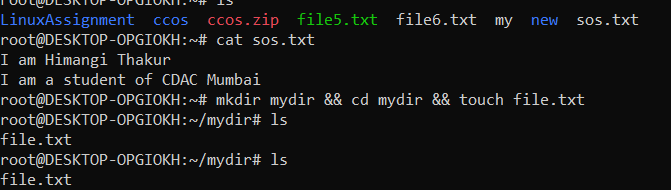


 kill PID gracefully terminates a process.

 kill -9 PID forcefully stops a process.

 pkill process\_name **kills all processes** with a specific name.

* mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

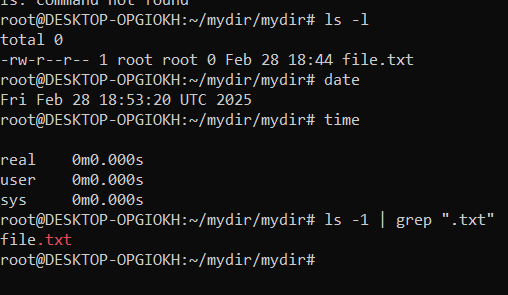


 Creates mydir directory.

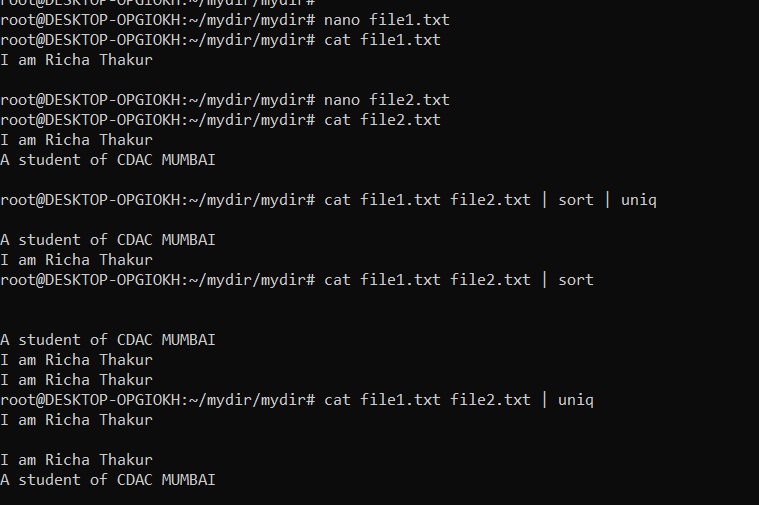
 Moves into mydir.

 Creates an empty file.txt inside mydir.

* ls -l | grep ".txt"



* cat file1.txt file2.txt | sort | uniq



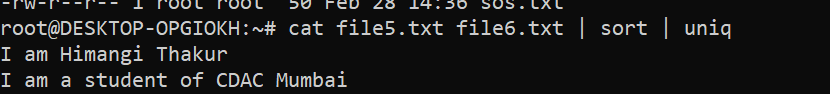
* ls –l | grep “^d”

Lists only directories

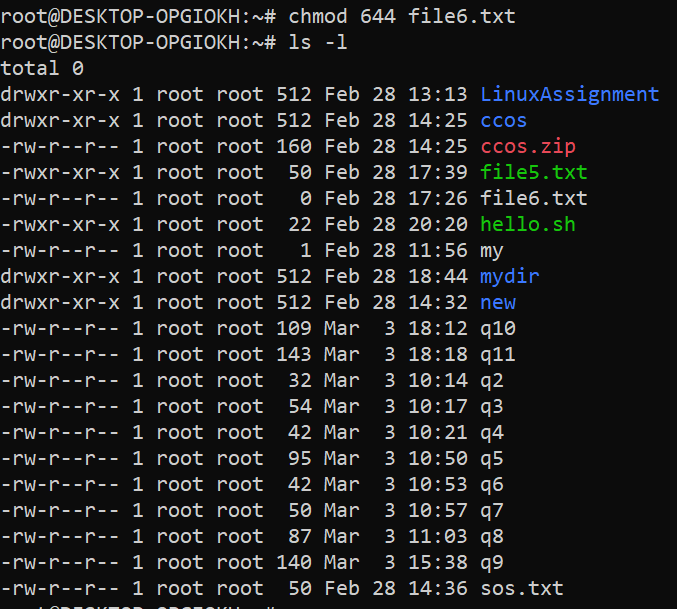
* grep -r "pattern" /path/to/directory/

Recursively searches for “pattern” in all files under the directory.

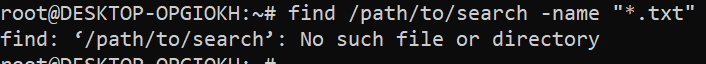
* cat file1.txt file2.txt | sort | uniq –d



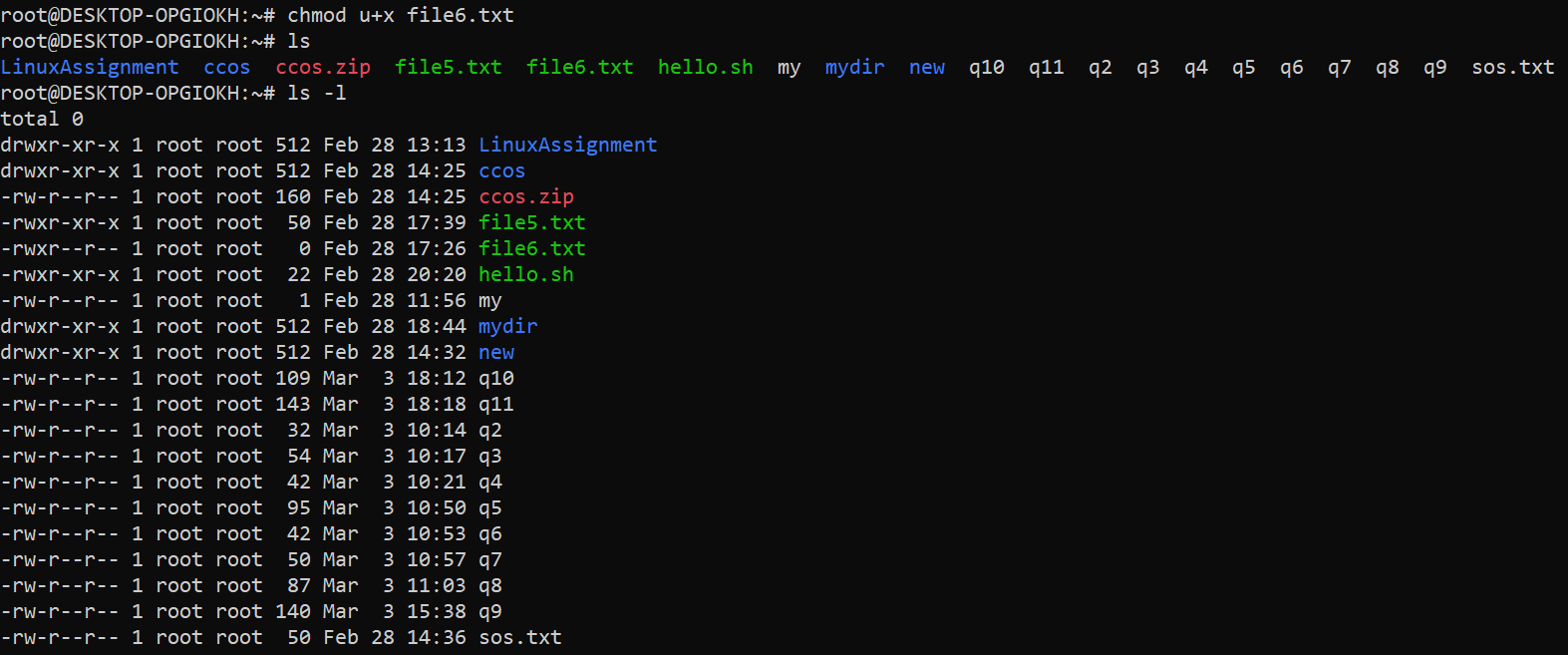
* chmod 644 file.txt



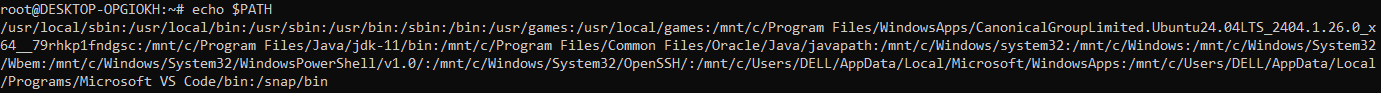
* cp -r source\_directory destination\_directory
* Recursively copies a directory and all its contents to anew location.
* find /path/to/search -name "\*.txt"



* chmod u+x file.txt



* echo $PATH



***Part B***

**Identify True or False**:

1**. ls** is used to list files and directories in a directory. True

2. **mv** is used to move files and directories. True

3. **cd** is used to copy files and directories. False

4. **pwd** stands for "print working directory" and displays the current directory. True

5**. grep** is used to search for patterns in files. True

6. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. True

7**. mkdir** -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist. True

8. **rm -rf file.txt** deletes a file forcefully without confirmation. True

**Identify the Incorrect Commands:**

1. **chmodx** is used to change file permissions. Incorrect

2. **cpy** is used to copy files and directories. Incorrect

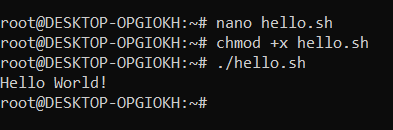
3. **mkfile** is used to create a new file. Incorrect

4. **catx** is used to concatenate files. Incorrect

5. **rn** is used to rename files. Incorrect

***Part C***

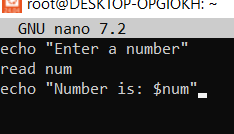
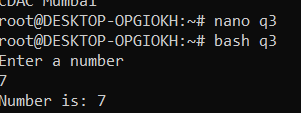
**Question 1:** Write a shell script that prints "Hello, World!" to the terminal.



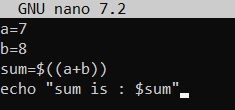
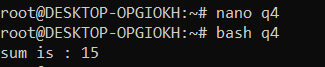
**Question 2:** Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



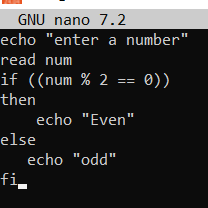
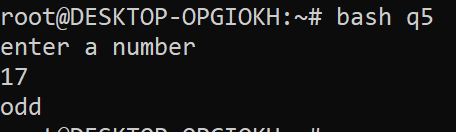
**Question 3:** Write a shell script that takes a number as input from the user and prints it.

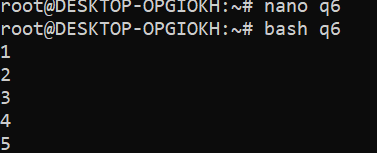
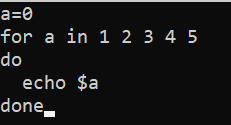
**Question 4**: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

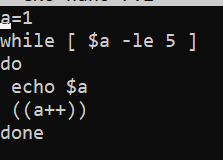
**Question 5:** Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

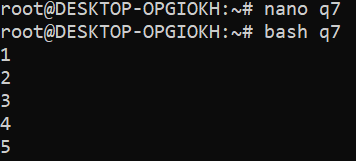
 

**Question 6:** Write a shell script that uses a for loop to print numbers from 1 to 5.

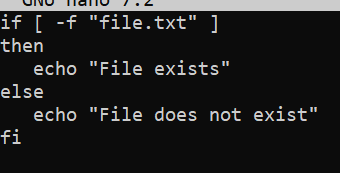


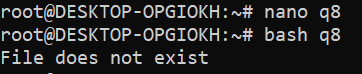
**Question 7:** Write a shell script that uses a while loop to print numbers from 1 to 5.



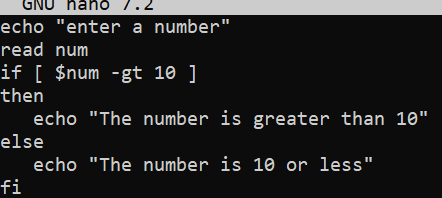


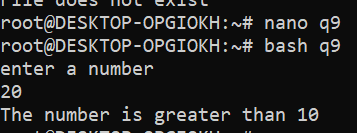
**Question 8:** Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



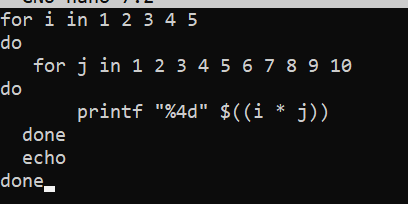
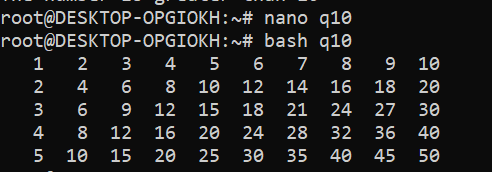


**Question 9:** Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

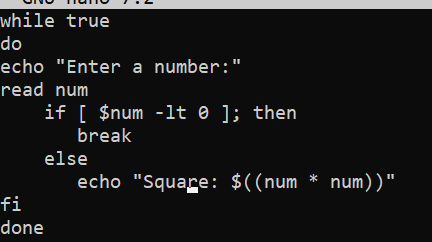
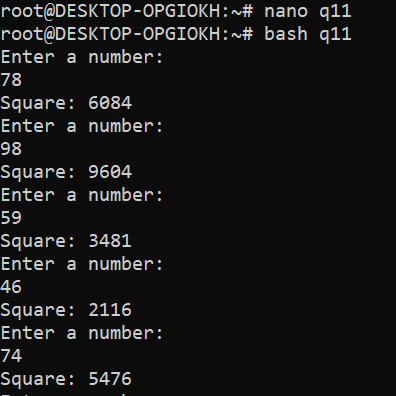




**Question 10:** Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

**Question 11:** Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered

***Part E***

1. Consider the following processes with arrival times and burst times:

| Process | Arrival Time | Burst Time |

|---------|--------------|------------|

| P1 | 0 | 5 |

| P2 | 1 | 3 |

| P3 | 2 | 6 |

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

--Process | Arrival Time | BT | RT | WT | TAT |

P1 0 | 5 | 0 0 5

P2 1 3 5 4 7

P3 2 6 8 6 12

AVG wating--- 0 + 4 + 6

3

= 3.33

Gantt Chart ---- P1 | P2 | P3 |

0 | 5 | 8 | 14

2. Consider the following processes with arrival times and burst times:

| Process | Arrival Time | Burst Time |

|---------|--------------|------------|

| P1 | 0 | 3 |

| P2 | 1 | 5 |

| P3 | 2 | 1 |

| P4 | 3 | 4 |

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

* Process | Arrival Time | BT | RT | WT | TAT |

P1 | 0 | 3 | 0 | 0 | 3

P2 | 1 | 5 | 8 | 7 | 12

P3 | 2 | 1 | 3 | 1 | 2

P4 | 3 | 4 | 4 | 1 | 5

AVg TAT---- 3+12+2+5 = 5.5

4

Gantt Chart p1 | p3 | p4 | p2 |

0 | 3 | 4 | 8 | 13

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

| Process | Arrival Time | Burst Time | Priority |

|---------|--------------|------------|----------|

| P1 | 0 | 6 | 3 |

| P2 | 1 | 4 | 1 | 2

| P3 | 2 | 7 | 4 |

| P4 | 3 | 2 | 2 |

Calculate the average waiting time using Priority Scheduling.

| Process | Arrival Time | Burst Time | priority | RT | WT

| P1 | 0 | 6 | 3 | 0 | 0+6= 6

| P2 | 1 | 4 | 1 | 1 | 0

| P3 | 2 | 7 | 4 | 12 | 10

| P4 | 3 | 2 | 2 | 5 | 2

Avg WT= 6+0+10+2 = 4.5

4

Gantt Chart p1 | p2 | p4 | p1 | P3

0 | 1 | 5 | 7 | 12 | 16

4. Consider the following processes with arrival times and burst times, and the time quantum for

Round Robin scheduling is 2 units:

| Process | Arrival Time | Burst Time |

| P1 | 0 | 4 |

| P2 | 1 | 5 |

| P3 | 2 | 2 |

| P4 | 3 | 3 |

Calculate the average turnaround time using Round Robin scheduling.

| Process | Arrival Time | Burst Time | RT | WT | CP | TAT

| P1 | 0 | 4 | 0 | 0+6=6 | 10 | 10

| P2 | 1 | 5 | 2 | 1+6+1=8 | 14 | 13

| P3 | 2 | 2 | 4 | 2 | 6 | 4

| P4 | 3 | 3 | 6 | 3+4=7 | 13 | 10

Avg TAT = 10+13+4+10 = 37 = 9.25

4 4

Gantt Chart p1 p2 p3 p4 p1 p2 p4 p2

0 2 4 6 8 10 12 14 16

5. Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?

#### Before forking:

Parent process: x = 5

Afterfork():

* The child process gets a separate copy of x = 5.
* Both parent and child increment x by 1 separately.

Final values of x:

| process | Initial x | After Increment (x + 1) |
| --- | --- | --- |
| Parent | 5 | 6 |
| Child | 5 | 6 |