CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  ----Prints Hello World
- name="Productive"
  ----craetes a variable ‘name’ and stores its value as ‘productive’
- touch file.txt
  -----creates file named file.txt
- ls -a
  ------ list all files along with the hidden ones
- rm file.tx
  ------removes file named file.txt
- cp file1.txt file2.txt
  ------ copies content of file1.txt to file2.txt
- mv file.txt /path/to/directory/
  ------ moves file.txt into the given directory
- chmod 755 script.sh
  ---- sets file permission as rwx for owner, rx for group and other users
- grep "pattern" file.tx
  ------ searches for word “pattern” inside file.txt
- kill PID
  ------terminates the process with that process ID
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  ------creates directory then goes into the created directory then creates a file named file.txt,then writes Hello World! and displays it using cat command
- ls -l | grep ".txt"
  -----lists all files having .txt extension along with their permissions
- cat file1.txt file2.txt | sort | uniq
  ------combines files file1.txt and file2.txt then sorts and gives unigue content
- ls -l | grep "^d"
  ------- lists only directory file type
- grep -r "pattern" /path/to/directory/
  ----- searches for “pattern” recursively
- cat file1.txt file2.txt | sort | uniq –d
  ----- show only duplicates after combing file1.txt and file2.txt
- chmod 644 file.txt -------rw permissions for owner and r permission for group and others
- cp -r source_directory destination_directory -------copies directory
- find /path/to/search -name "*.txt" -----finds all .txt files at the mentioned location
- chmod u+x file.txt ----- grants permission to execute file.txt to the user
- echo $PATH--------displays system path for executables

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 (cd is for changing directory)
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
2. **cpy** is used to copy files and directories.
3. **mkfile** is used to create a new file.
4. **catx** is used to concatenate files.
5. **rn** is used to rename file

==> Here all 5 command sare wrong

1. chmod is used to change file permissions
2. cp is used to copy files and directories
3. touch is used to create new files
4. cat is used to concatenate files
5. mv is used to rename files

Part C

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

```
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ vi a21.txt
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ chmod u+x a21.txt
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ ls -l
total 36
-rwxr--r-- 1 cdac cdac   34 Aug 21 17:58 a21.txt
-rw-r--r-- 1 cdac cdac  313 Aug 21 16:25 data.txt
drwxr-xr-x 2 cdac cdac 4096 Aug 20 16:19 docs
-rw-r--r-- 1 cdac cdac   44 Aug 21 16:42 duplicate.txt
-rw-r--r-- 1 cdac cdac   30 Aug 19 13:14 file1.txt
-rw-r--r-- 1 cdac cdac   54 Aug 21 16:45 fruit.txt
-rw-r--r-- 1 cdac cdac  266 Aug 21 16:38 input.txt
-rw-r--r-- 1 cdac cdac   54 Aug 21 16:28 numbers.txt
-rw-r--r-- 1 cdac cdac  266 Aug 21 16:39 output.txt
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ ./a21.txt
Hello, World!
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$
```

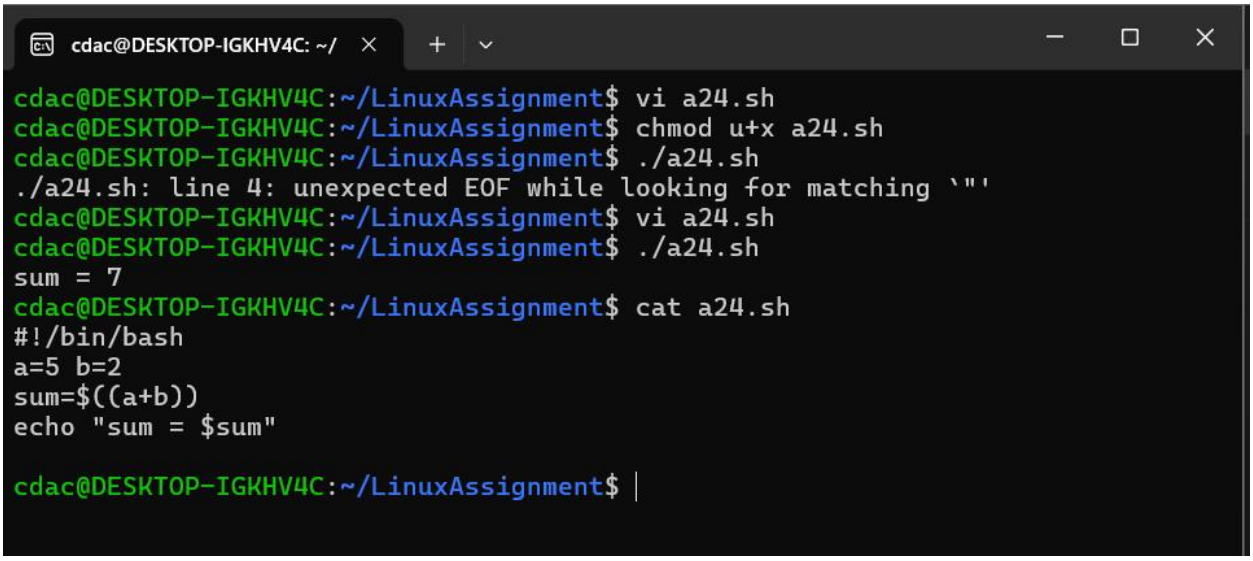
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

```
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ vi a22.sh
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ chmod u+x a22.sh
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ ./a22.sh
CDAC MUMBAI
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ cat a22.sh
#!/bin/bash
name="CDAC MUMBAI"
echo $name

cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$
```

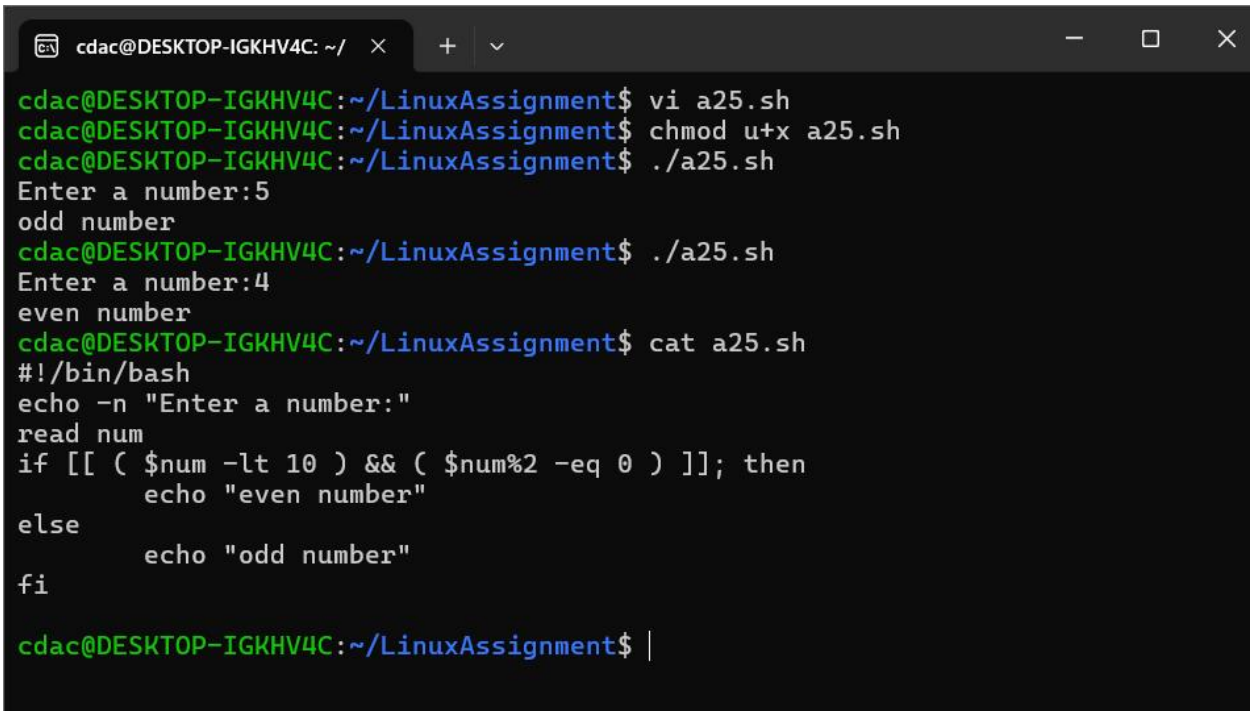
Question 3: Write a shell script that takes a number as input from the user and prints it.

```
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ vi a23.sh
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ chmod u+x a23.sh
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ ./a23.sh
Enter a number: 55
You entered: 55
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$ cat a23.sh
#!/bin/bash

read -p "Enter a number: " num
echo "You entered: $num"
cdac@DESKTOP-IGKHV4C:~/LinuxAssignment$
```

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.

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a26.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a26.sh
cdac@DESKTOP-IGKHV4C:~$ ./a26.sh
1
2
3
4
5
cdac@DESKTOP-IGKHV4C:~$
```

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

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a27.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a27.sh
cdac@DESKTOP-IGKHV4C:~$ ./a27.sh
1
2
3
4
5
cdac@DESKTOP-IGKHV4C:~$ cat a27.sh
#!/bin/bash
i=1
while [ $i -le 5 ]
do
  echo $i
  ((i++))
done

cdac@DESKTOP-IGKHV4C:~$
```

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".

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a28.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a28.sh
cdac@DESKTOP-IGKHV4C:~$ ./a28.sh
File does not exist
cdac@DESKTOP-IGKHV4C:~$ cat a28.sh
#!/bin/bash
if [ -f "file.txt" ]; then
  echo "File exists"
else
  echo "File does not exist"
fi

cdac@DESKTOP-IGKHV4C:~$
```

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.

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a29.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a29.sh
cdac@DESKTOP-IGKHV4C:~$ ./a29.sh
Enter a number: 11
Greater than 10
cdac@DESKTOP-IGKHV4C:~$ ./a29.sh
Enter a number: 9
Not greater than 10
cdac@DESKTOP-IGKHV4C:~$ cat a29.sh
#!/bin/bash

read -p "Enter a number: " n
if [ $n -gt 10 ]; then
  echo "Greater than 10"
else
  echo "Not greater than 10"
fi

cdac@DESKTOP-IGKHV4C:~$
```

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.

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a210.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a210.sh
cdac@DESKTOP-IGKHV4C:~$ ./a210.sh
   1   2   3   4   5
   2   4   6   8  10
   3   6   9  12  15
   4   8  12  16  20
   5  10  15  20  25
cdac@DESKTOP-IGKHV4C:~$ cat a210.sh
#!/bin/bash
for i in {1..5}
do
  for j in {1..5}
  do
    printf "%4d" $((i*j))
  done
  echo
done

cdac@DESKTOP-IGKHV4C:~$
```

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.

```
cdac@DESKTOP-IGKHV4C: ~
cdac@DESKTOP-IGKHV4C:~$ vi a211.sh
cdac@DESKTOP-IGKHV4C:~$ chmod u+x a211.sh
cdac@DESKTOP-IGKHV4C:~$ ./a211.sh
Enter a number: 3
Square = 9
Enter a number: 4
Square = 16
Enter a number: 5
Square = 25
Enter a number: -1
cdac@DESKTOP-IGKHV4C:~$ cat a211.sh
#!/bin/bash

while true
do
  read -p "Enter a number: " n
  if [ $n -lt 0 ]; then
    break
  fi
  echo "Square = $((n*n))"
done
cdac@DESKTOP-IGKHV4C:~$
```

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.

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.

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
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

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.

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?

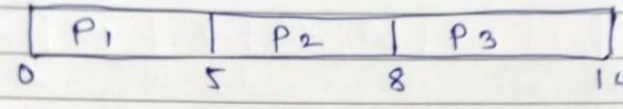
Assignment 2

PART E

Q.1

Process	AT	BT
P_1	0	5
P_2	1	3
P_3	2	6

→ Using FCFS, find avg. waiting time.



⇒ Waiting time -

P_1 = Start time - Arrival time

= 0 − 0

$P_1 = 0$

$P_2 = 5 - 1 = 4$

$P_3 = 8 - 2 = 6$

Avg. waiting time = $\frac{0+4+6}{3} = 3.33$

Q.2

Process	AT	BT
P_1	0	3
P_2	1	5
P_3	2	1
P_4	3	4

Calculate avg. TAT using SJF.

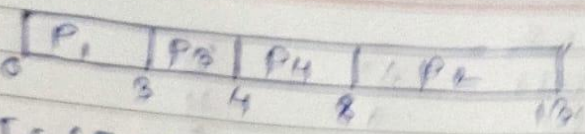
→ TAT = BT − AT

⇒ TAT for -

$P_1 = 3 - 0 = 3$

$P_2 = 5 - 1 = 4$

$P_3 = 1 - 2$



TAT = CT - AT

For, $P_1 = 3 - 0 = 3$

$P_3 = 4 - 2 = 2$

$P_4 = 8 - 3 = 5$

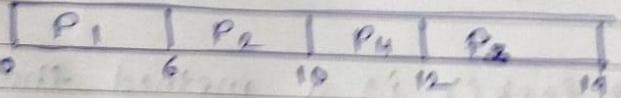
$P_2 = 13 - 1 = 12$

Avg. TAT = $\frac{3+2+5+12}{4} = 5.5$

Q5

Process	AT	BT	Priority (lower is higher)
P_1	0	6	3
P_2	1	4	1
P_3	2	7	4
P_4	3	2	2

→



Waiting time = TAT - BT

$P_1 = (6 - 0) - 6 = 0$

$P_2 = (10 - 1) - 4 = 5$

$P_4 = (12 - 3) - 2 = 7$

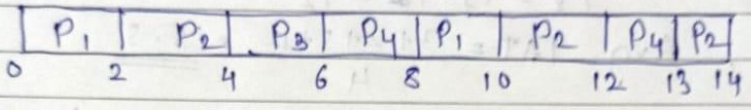
$P_3 = (19 - 2) - 7 = 10$

Avg. WT = $\frac{0+5+7+10}{4} = 5.5$

Q.4. TQ = 2

Process	AT	BT
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	3

⟶



TAT = CT - AT

⇒ TAT for, $P_1 = 10 - 0 = 10$

$P_2 = 14 - 1 = 13$

$P_3 = 6 - 2 = 4$

$P_4 = 13 - 3 = 10$

Avg. TAT = $\frac{10+13+4+10}{4} = 9.25$

Q.5.

⟶

- fork() is a method that creates a copy of the parent process.
- Each process has its own separate copy of variable x.
- When in both process's increment of x by 1, it becomes 6.
- Each process will get value of 6.