CDAC MUMBAI

Concepts of Operating System

Assignment 2

Name:	Sumit Prabhakar Mote
Course:	PG-DAC
Batch:	August 2025
Subject:	Concepts of Operating Systems

Part A

What will the following commands do?

1. echo "Hello, World!"
   Ans:
   This command will print Hello World.

2. Name= “Productive”
   Ans:
   The Productive value is assigning to a variable called Name.

3. touch file.txt
   Ans:
   touch command helps to create a new file called file.txt

4. ls -a
   Ans:
   It will list all files and directories, including hidden ones, within the current directory.

5. rm file.txt

   Ans:

   The file.txt will get removed/deleted.

6. cp file1.txt file2.txt

   Ans:

   The content of file1.txt file will get copied and paste to the destination file file2.txt

7. mv file.txt /path/to/directory/

   Ans:

   The mv file.txt /path/to/directory/ command is used move a file from its current location to a specified destination directory.

8. chmod 755 script.sh

   Ans:

   chmod is a command which change the permission of a file. 755 is a octal number that represents the new permissions.
   here 1st number(7) gives all permission (read,write, execute) to Owner/user. 2nd and 3rd number (5 and 5) gives read and execute permission to group and other users.

9. grep “pattern” file.txt

   Ans:

   grep is command is use to search for a content in a file. It will search for “pattern” word in file1.txt.

10. kill PID

    Ans:

    The command kill PID is used to terminate a process.

11. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
    Ans:
    This command creates a new directory named mydir, changes the current directory to mydir, creates an empty file named file.txt, writes the text "Hello, World!" into file.txt, and then displays the content of file.txt to the terminal.

12. ls -l | grep ".txt"
    Ans:
    Lists all files and directories in the current folder and finds all the lines that end in .txt.

13. cat file1.txt file2.txt | sort | uniq
    Ans:
    Combines the contents of file1.txt and file2.txt, sorts all the lines, and then removes any duplicate lines.

14. chmod 644 file.txt
    Ans:
    Changes the file permissions so the owner can read and write, while others can only read.

15. cp -r source_directory destination_directory
    Ans:
    Copies an entire directory and all of its contents to a new location.

16. find /path/to/search -name "*.txt"
    Ans:
    Searches for all files ending with .txt in a specified directory and its subdirectories.

17. chmod u+x file.txt
    Ans:
    Adds the execute permission to a file for its owner, making the file runnable as a program.

18. Echo $path
    Ans:
    Prints the list of directories where the system looks for executable commands.

Part B

Identify True or False:

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

2. mv is used to move files and directories.
   Ans: True

3. cd is used to copy files and directories.
   Ans: False

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

5. grep is used to search for patterns in files.
   Ans: False

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

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

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

Part C

Question 1:

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

Ans:

```
cdac@sumitideapad:~$ vi Hello.sh
cdac@sumitideapad:~$ chmod +x Hello.sh
cdac@sumitideapad:~$ ./Hello.sh
Hello, World!
cdac@sumitideapad:~$ cat Hello.sh
echo "Hello, World!"
cdac@sumitideapad:~$
```

Question 2:

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

Ans:

```
cdac@sumitideapad:~$ vi print.sh
cdac@sumitideapad:~$ chmod +x print.sh
cdac@sumitideapad:~$ ./print.sh
CDAC Mumbai
cdac@sumitideapad:~$ cat print.sh
name="CDAC Mumbai"
echo $name

cdac@sumitideapad:~$
```

Question 3:

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

Ans:

```
cdac@sumitideapad:~$ vi input.sh
cdac@sumitideapad:~$ chmod +x input.sh
cdac@sumitideapad:~$ ./input.sh
Enter a Number:
7
The given Number is 7
cdac@sumitideapad:~$ cat input.sh
echo "Enter a Number: "
read Number
echo "The given Number is $Number"
cdac@sumitideapad:~$
```

Question 4:

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

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi addition.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x addition.sh
cdac@sumitideapad:~/Shell_Scripts$ ./addition.sh
 Enter First Number: 10
 Enter Second Number: 15
 25 is the addition.
cdac@sumitideapad:~/Shell_Scripts$ cat addition.sh
#!/bin/bash
echo -n " Enter First Number: "
read x
echo -n " Enter Second Number: "
read y
((Sum= x+y))
echo " $Sum is the addition."
cdac@sumitideapad:~/Shell_Scripts$
```

Question 5:

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

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi EvenOdd.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x EvenOdd.sh
cdac@sumitideapad:~/Shell_Scripts$ ./EvenOdd.sh
Enter a Num:
5
5 is Odd.
cdac@sumitideapad:~/Shell_Scripts$ ./EvenOdd.sh
Enter a Num:
8
8 is Even.
cdac@sumitideapad:~/Shell_Scripts$ cat EvenOdd.sh
#!/bin/bash
echo "Enter a Num:"
read Num

rem=$((Num % 2))
if [ "$rem" -eq 0 ]
then
        echo "$Num is Even."
else
        echo "$Num is Odd."
fi

cdac@sumitideapad:~/Shell_Scripts$
```

Question 6:

Write a shell script that uses a for loop to print numbers from 1 to 5.

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi loop.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x loop.sh
cdac@sumitideapad:~/Shell_Scripts$ ./loop.sh
1
2
3
4
5
cdac@sumitideapad:~/Shell_Scripts$ cat loop.sh
#!/bin/bash

for i in {1..5}
do
    echo $i
done
cdac@sumitideapad:~/Shell_Scripts$
```

Question 7:

Write a shell script that uses a while loop to print numbers from 1 to 5

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi loop.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x loop.sh
cdac@sumitideapad:~/Shell_Scripts$ ./loop.sh
1
2
3
4
5
cdac@sumitideapad:~/Shell_Scripts$ cat loop.sh
#!/bin/bash

count=1
while [ $count -le 5 ]
do
  echo $count
  count=$((count + 1))
done
cdac@sumitideapad:~/Shell_Scripts$
```

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".

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi check.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x check.sh
cdac@sumitideapad:~/Shell_Scripts$ ./check.sh
File does not exist
cdac@sumitideapad:~/Shell_Scripts$ touch file.txt
cdac@sumitideapad:~/Shell_Scripts$ ./check.sh
File exists
cdac@sumitideapad:~/Shell_Scripts$ cat check.sh
#!/bin/bash

if [ -f "file.txt" ]
then
  echo "File exists"
else
  echo "File does not exist"
fi
cdac@sumitideapad:~/Shell_Scripts$
```

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.

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi greaterthan.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x greaterthan.sh
cdac@sumitideapad:~/Shell_Scripts$ ./greaterthan.sh
Enter a Number:
15
The Number is greater than 10.
cdac@sumitideapad:~/Shell_Scripts$ ./greaterthan.sh
Enter a Number:
6
The Number is not greater than 10.
cdac@sumitideapad:~/Shell_Scripts$ cat greaterthan.sh
#!/bin/bash

echo "Enter a Number: "
read number

if [ "$number" -gt 10 ]
then
    echo "The Number is greater than 10."
else
    echo "The Number is not greater than 10."
fi
cdac@sumitideapad:~/Shell_Scripts$
```

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.

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi multiplication.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x multiplication.sh
cdac@sumitideapad:~/Shell_Scripts$ ./multiplication.sh
Multiplication Table (1-5)
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@sumitideapad:~/Shell_Scripts$ cat multiplication.sh
#!/bin/bash

echo "Multiplication Table (1-5)"
for i in {1..5}
do
  for j in {1..5}
  do
    result=$((i*j))
    echo -ne "$result\t"
  done
  echo ""
done
cdac@sumitideapad:~/Shell_Scripts$
```

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.

Ans:

```
cdac@sumitideapad:~/Shell_Scripts$ vi q11.sh
cdac@sumitideapad:~/Shell_Scripts$ chmod +x q11.sh
cdac@sumitideapad:~/Shell_Scripts$ ./q11.sh
Enter a positive number (or a negative number to exit):
5
The square of 5 is: 25
Enter a positive number (or a negative number to exit):
4
The square of 4 is: 16
Enter a positive number (or a negative number to exit):
8
The square of 8 is: 64
Enter a positive number (or a negative number to exit):
19
The square of 19 is: 361
Enter a positive number (or a negative number to exit):
-5
Negative number entered.Thank you..
Script finished.
cdac@sumitideapad:~/Shell_Scripts$ cat q11.sh
#!/bin/bash
while true
do
  echo "Enter a positive number (or a negative number to exit): "
  read number
  if [ "$number" -lt 0 ]
  then
    echo "Negative number entered.Thank you.."
    break
  fi
  square=$((number * number))
  echo "The square of $number is: $square"
done
echo "Script finished."
cdac@sumitideapad:~/Shell_Scripts$
```

Part D

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?

Ans:

Assignment 2

Numericals.

Q1) Consider the following processes with arrival time & burst times:

Process	Arrival time	Burst time
P_1	0	5
P_2	1	3
P_3	2	6

Calculate average waiting time using First-come, first-serve Scheduling.

→

Start arrival

P_1 = Waiting time: 0 - 0 = 0
P_2 = Waiting time: 5 - 1 = 4
P_3 = Waiting time: 8 - 2 = 6

Total waiting time = 10

Average waiting time = 3.33 //

Q2) Consider the following data:

Process	Arrival time	Burst time
P_1	0	3
P_2	1	5
P_3	2	1
P_4	3	4

Calculate average turnaround time using SJF Scheduling.

→

P_1 = Turnaround time : 3 - 0 = 3
P_3 = TT : 4 - 2 = 2
P_4 = TT : 8 - 3 = 5
P_2 = TT : 13 - 1 = 12

Total Turnaround time = 22

Average turnaround time = 5.5

Q3) Consider the following data:

Process	Arrival time	Burst time	Priority
P_1	0	6	3
P_2	1	4	1
P_3	2	7	4
P_4	3	2	2

Calculate average waiting time using Priority Scheduling.

→

P_1 : Waiting time = (1-0) + (5-1) = 4
P_2 = Wt = (3-1) + (7-3) = 6
P_3 = Wt = 7 - 2 = 5
P_4 = Wt = 3 - 3 = 0

Total average waiting time = 15

Average = 15 ÷ 4 = 3.75 //

Q4) Consider the following table.

Process	Arrival time	Burst time
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	3

Calculate the average turnaround time using Round Robin scheduling.

→

P_1 = turnaround time = 10 - 0 = 10
P_2 = tt = 14 - 1 = 13
P_3 = tt = 6 - 2 = 4
P_4 = tt = 13 - 3 = 10

Total = 37

Average = 37/4 = 9.25 //

Q5) Consider a program that uses fork() system call to create child process. Initially, the parent process ~~have~~ has a variable x with value 5. After forking, both processes increment the value of x by 1. What will be the final values of x after fork() call?

→

- fork() creates a copy of the parent process.
- Each process has its own seperate copy of variable x.
- When both process increment x by 1, it becomes 6.
- Each process will get value of 6.