

# Concepts of Operating System

## Assignment 2

### Part A

1 What will the following commands do?

- `echo "Hello, World!"`

Ans : It will Print Hello World.

- `name="Productive"`

Ans:

- `touch file.txt`

Ans: Touch command is used to create new file.

- `ls -a`

Ans: using this command list of all files ,directories and bash history.

- `rm file.txt`

Ans: This command is used to remove or delete a file.

- `cp file1.txt file2.txt`

Ans: This command is used to copy the content of file1.txt to file2.txt.

- `mv file.txt /path/to/directory/`

Ans: This command is used to move file from one directory to another by giving path .

- `chmod 755 script.sh`

Ans: This change mode command is used to give owner permission for read, write, execute of an file or directory.

- `grep "pattern" file.txt`

Ans: This command is used search for specific pattern in the file.

- kill PID

Ans: The kill PID command is used to completely delete that process from system. This method is used when deadlock occurs.

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

Ans: First make directory having name mydir, second change current directory to mydir ,third by using touch command create anew file having name file.txt, fourth write hello world program in file.txt and fifth print that echo statement Hello World using cat command.

- ls -l | grep ".txt"

Ans: print the all list of file and directiores by sorting that files with txt pattern.

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

Ans: By using piping method sort the file1.txt and file2.txt in alphabetical order ,and display that sorted file output by using cat command.

- ls -l | grep "^d"

Ans: This command is used to show all directories in current directory.

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

Ans: This command is used to recursively search for specific pattern in given defined path in defined directory.

- cat file1.txt file2.txt | sort | uniq -d

Ans: In this file1.txt and file2.txt are sorted using piping . the duplicate data will be displayed using cat command.

- chmod 644 file.txt

Ans: This command gives the permission to read and write but other user can only read this file .

- cp -r source\_directory destination\_directory

Ans: Copies the all files to another folder of another directory.

- `find /path/to/search -name "*.txt"`

Ans: Search the file having .txt extension in given path .

- `chmod u+x file.txt`

Ans: This command allow to change permission of user to execute that file.

- `echo $PATH`

Ans: It will show the path of bash shell.

## **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: True.

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: False.

Identify the Incorrect Commands:

1. `chmodx` is used to change file permissions.

Ans: `chmod`/option/mod/filename.

2. `cpy` is used to copy files and directories.

Ans : `cp filename1 filename2`.

3. `mkfile` is used to create a new file.

Ans: `touch filename.txt`.

4. `catx` is used to concatenate files.

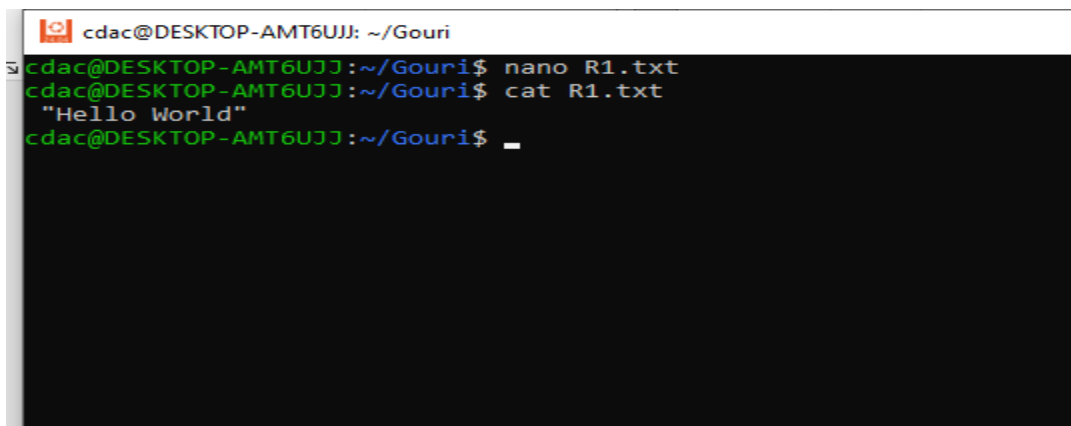
Ans: `Cat filename`.

5. `rn` is used to rename files.

Ans: `rn file.txt file1.txt`

## Part C

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



```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano R1.txt
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat R1.txt
"Hello World"
cdac@DESKTOP-AMT6UJJ:~/Gouri$
```

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

```
Select cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh1
cdac@DESKTOP-AMT6UJJ:~/Gouri$
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh1
name="CDAC Mumbai"
echo " $name"
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh1
CDAC Mumbai
cdac@DESKTOP-AMT6UJJ:~/Gouri$ _
```

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

```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh1
cdac@DESKTOP-AMT6UJJ:~/Gouri$
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh1
name="CDAC Mumbai"
echo " $name"
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh1
CDAC Mumbai
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh1
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh1

cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh1 10 20 25 50 99
10
20
25
50
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh1
echo $1
echo $2
echo $3
echo $4
cdac@DESKTOP-AMT6UJJ:~/Gouri$ _
```

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

```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh2
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh2
a=0
sum=0
for a in 5 3
do
echo $a
sum=`expr $sum + $a`
done
echo Sum is, $sum
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh2
5
3
Sum is, 8
cdac@DESKTOP-AMT6UJJ:~/Gouri$ _
```

result.

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

```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh3
cdac@DESKTOP-AMT6UJJ:~/Gouri$ y
y: command not found
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh3
Enter a number
5
n Result:
sh3: line 5: [1: command not found
5 is odd number
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh3
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh3
Enter a number
5
n Result:
sh3: line 5: [1: command not found
5 is odd number
cdac@DESKTOP-AMT6UJJ:~/Gouri$ _
```

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

```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh4
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh4
1
2
3
4
5
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh4
a=0
for a in 1 2 3 4 5
do
echo $a
done
cdac@DESKTOP-AMT6UJJ:~/Gouri$
```

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".

```
cdac@DESKTOP-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh8
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh8
File is not exist
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh8
if test -f "file.txt" ;
then
echo "File is exist"
else
echo "File is not exist"
fi
cdac@DESKTOP-AMT6UJJ:~/Gouri$
```

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-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano sh4
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash sh4
1
2
3
4
5
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat sh4
a=0
for a in 1 2 3 4 5
do
echo $a
done
cdac@DESKTOP-AMT6UJJ:~/Gouri$
```

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-AMT6UJJ: ~/Gouri
cdac@DESKTOP-AMT6UJJ:~/Gouri$ nano shell7
cdac@DESKTOP-AMT6UJJ:~/Gouri$ bash shell7
Enter a number
4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
cdac@DESKTOP-AMT6UJJ:~/Gouri$ cat shell7
echo "Enter a number"
read n
i=1
while [ $i -lt 10 ]
do
result=`expr $i \* $n`
echo "$n * $i = $result"
((i++))
done
cdac@DESKTOP-AMT6UJJ:~/Gouri$
```



## Part E

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

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

Process Id	Arrival Time	Burst Time	Response Time	Waiting Time	Turn Around Time
P1	0	5			
P2	1	3			
P3	2	6			

Ans:

Gantt Chart= 0-----5-----8 -----14

P1      P2      P3

Waiting Time= Response Time – Arrival Time

For P1      Waiting Time= 0-0=0

For P2      Waiting Time=5-1=4

For P3      Waiting Time=8-2=6

Turn Around Time= Waiting Time + Burst Time

For P1 TAT= 0+5=5

For P2 TAT= 4+3=7

For P3 TAT=6+6=12

Average Response Time =(0+5+8)/3=4.33

Average Waiting Time=(0+4+6)/3=3.33

Average TAT=(5+7+12)/3=8

Process Id	Arrival Time	Burst Time	Response Time	Waiting Time	Turn Around Time
P1	0	5	0	0	5
P2	1	3	5	4	7
P3	2	6	8	6	12
			Average =4.33	Average=3.33	Average=8

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

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

Process ID	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Ans:

Gantt Chart= 0-----1-----4-----8-----13  
                  P3      P1      P4      P2

Waiting Time= Response Time – Arrival Time

For P1      Waiting Time= 1-0=1

For P2      Waiting Time=8-1=7

For P3      Waiting Time=0-2=2

For P4      Waiting Time=4-3=1

Turn Around Time= Waiting Time + Burst Time

For P1 TAT= 1+3=4

For P2 TAT= 7+5=12

For P3 TAT=2+1=3

For P4 TAT=1+4=5

Average Response Time = $(1+8+0+4)/4=3.25$

Average Waiting Time= $(1+7+2+1)/4=2.75$

Average TAT= $(4+12+3+5)/4=6$

Process Id	Arrival Time	Burst Time	Response Time	Waiting Time	Turn Around Time
P1	0	3	1	1	4
P2	1	5	8	7	12
P3	2	1	0	2	3
P4	3	4	4	1	5
			Average =3.25	Average=2.75	Average=6

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

Calculate the average waiting time using Priority Scheduling.

Process ID	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Ans:

Gantt Chart= 0-----1-----4 -----6-----12-----19  
                   P2      P2      P4      P1      P3

Waiting Time= Resposnse Time – Arrival Time

For P1      Waiting Time= 6-0=6

For P2      Waiting Time=0-1=1

For P3      Waiting Time=12-2=10

For P4      Waiting Time=4-3=1

Turn Around Time= Waiting Time + Burst Time

For P1 TAT= 6+6=12

For P2 TAT=1+4=5

For P3 TAT=10+7=17

For P4 TAT=1+3=4

Average Response Time =(6+0+12+4)/4=5.5

Average Waiting Time=(6+1+10+1)/4=4.5

Average TAT=(12+5+17+3)/4=9.25

Process Id	Arrival Time	Burst Time	Priority	Response Time	Waiting Time	Turn Around Time
P1	0	6	3	6	6	12
P2	1	4	1	0	1	5
P3	2	7	4	12	10	17
P4	3	2	2	4	1	3
				Average=5.5	Average=4.5	Average=9.25

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

Calculate the average turnaround time using Round Robin scheduling.

ProcessID	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Ans:

Gantt Chart= 0-----2-----4-----6-----8-----10-----12-----14-----16  
                                 P1      P2          P3      P4      P1      P2      P4      P2

Waiting Time= Resposnse Time – Arrival Time

For P1      Waiting Time= 0-(0+6)=6

For P2      Waiting Time=1-(2+6)=7

For P3      Waiting Time=2-4=2

For P4      Waiting Time=3-(6+4)=7

Turn Around Time= Waiting Time + Burst Time

For P1 TAT= 6+4=10

For P2 TAT=7+5=12

For P3 TAT=2+2=4

For P4 TAT=7+3=10

Average Response Time =(0+2+4+6)/4=3

Average Waiting Time=(6+7+2+7)/4=5.5

Average TAT=(10+12+4+10)/4=9

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?