Concepts of Operating System- Assignment 2 March 1, 2025

Name: Shital Kawthe_DAC_KH

PART A

What will the following commands do?

1. echo "Hello, World!"

-Prints "Hello, World!" to the console.

2. name="Productive"

-Assigns the string "Productive" to a variable named "name".

3. touch file.txt

-Creates a new empty file named "file.txt".

4. ls -a

-Lists all files and directories in the current directory, including hidden ones.

5. rm file.txt

-Deletes the file "file.txt".

6. cp file1.txt file2.txt

-Copies the contents of "file1.txt" to a new file named "file2.txt".

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

-Moves the file "file.txt" to the specified directory.

8. chmod 755 script.sh

-Changes the permissions of the file "script.sh" to allow the owner to read, write, and execute, and the group and others to read and execute.

9. grep "pattern" file.txt

-Searches for the specified pattern in the file "file.txt" and prints the matching lines.

10. kill PID

-Terminates the process with the specified process ID (PID).

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

-Creates a new directory "mydir", navigates into it, creates a new file "file.txt", writes "Hello, World!" to it, and then prints the contents of the file.

12. ls -l | grep ".txt"

-Lists all files and directories in the current directory in a detailed format and searches for files with the ".txt" extension.

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

-Concatenates the contents of "file1.txt" and "file2.txt", sorts the output, and removes duplicate lines.

14. ls -l | grep "^d"

-Lists all files and directories in the current directory in a detailed format and searches for directories (which start with "d" in the output).

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

-Recursively searches for the specified pattern in all files within the specified directory and its subdirectories.

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

-Concatenates the contents of "file1.txt" and "file2.txt", sorts the output, and prints only the duplicate lines.

17. chmod 644 file.txt

-Changes the permissions of the file "file.txt" to allow the owner to read and write, and the group and others to read.

18. cp -r source_directory destination_directory

-Recursively copies the contents of the "source_directory" to the "destination_directory".

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

-Searches for files with the ".txt" extension within the specified directory and its subdirectories.

20. chmod u+x file.txt

-Adds execute permission for the owner of the file "file.txt".

21. echo $PATH

-Prints the value of the PATH environment variable, which lists the directories where the system searches for executable files.

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 change directories, not copy files and directories.

-False. The correct command for copying is cp.

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 directory and its contents forcefully without confirmation.

-False. To delete a file forcefully without confirmation, the correct command is rm -f file.txt.

PART C

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

echo "Hello, World!"

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

name="CDAC Mumbai"

echo "Name: $name"

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

echo "Enter a number: "

read num

echo "You entered: $num"

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

echo "Enter first number: "

read num1

echo "Enter second number: "

read num2

result=$((num1 + num2))

echo "The result is: $result"

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

read -p "Enter a number: " num

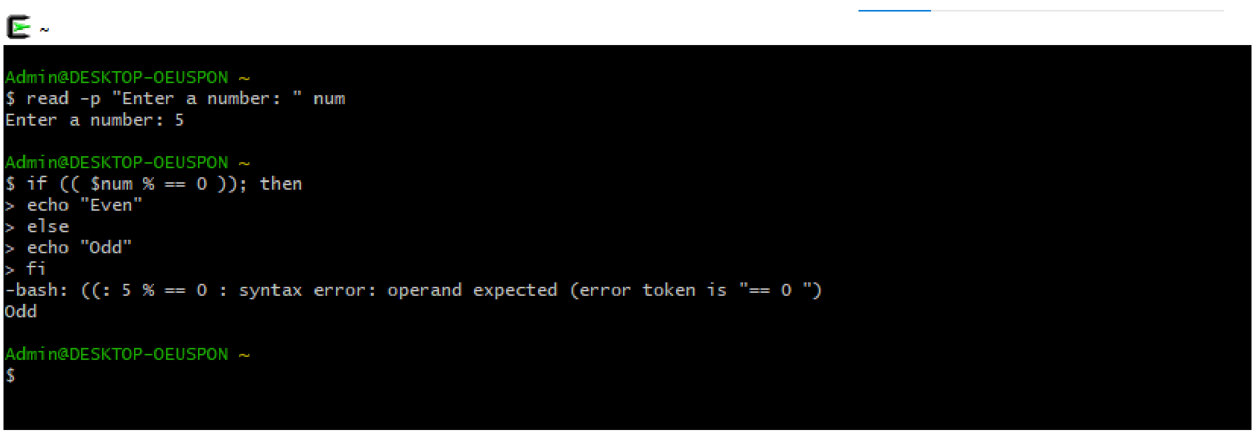
if (($num % 2 == 0)); then

  echo "Even"

else

  echo "Odd"

fi



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

for i in {1..5}; do

  echo $i

done

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

i=1

while [$i -le 5]; do

  echo $i

  ((i++))

Done

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".

Check if the file exists

if [-f "file.txt"]; then

  echo "File exists"

else

  echo "File does not exist"

fi

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.

num=15

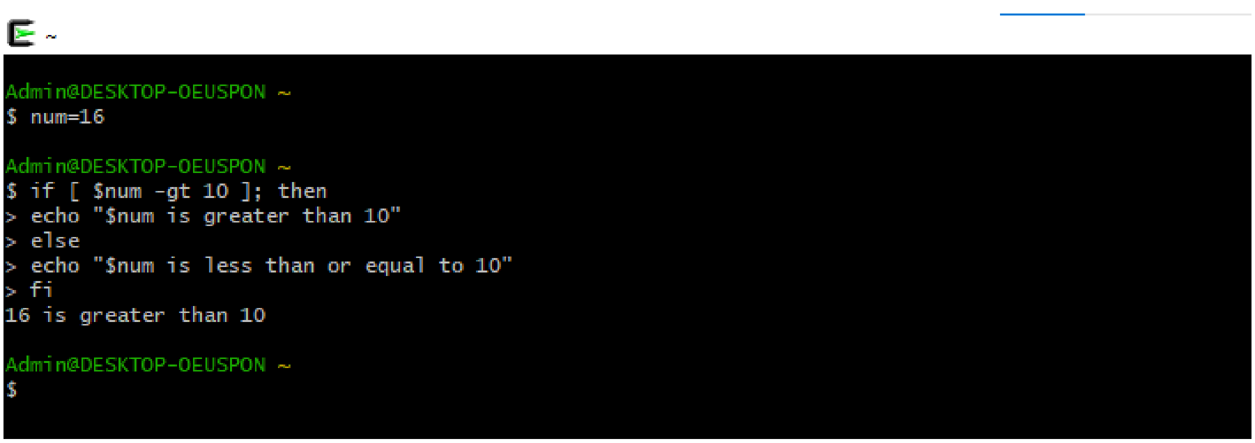
if [$num -gt 10]; then

  echo "$num is greater than 10"

else

  echo "$num is less than or equal to 10"

fi



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.

```
echo "Multiplication Table:"

for i in {1..5}; do

 for j in {1..5}; do

  printf "%4d" $((i*j))

 done

 echo

done
```

```
Admin@DESKTOP-OEUSPON ~
$ echo "Multiplication Table:"
Multiplication Table:

Admin@DESKTOP-OEUSPON ~
$ for i in {1,2,3,4,5,6,7}; do
> for j in {1,2,3,4,5,6,7}; do
> printf "%4d" $((i*j))
> done
> echo
> done
   1   2   3   4   5   6   7
   2   4   6   8  10  12  14
   3   6   9  12  15  18  21
   4   8  12  16  20  24  28
   5  10  15  20  25  30  35
   6  12  18  24  30  36  42
   7  14  21  28  35  42  49

Admin@DESKTOP-OEUSPON ~
$
```

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.

```
while true; do

 read -p "Enter a number: " num

 if [ $num -lt 0 ]; then

  break

 fi

 echo "Square of $num: $((num ** 2))"
```

done

~

```
Admin@DESKTOP-OEUSPON ~
$ while true; do
> read -p "Enter a number: " num
> if [ $num -lt 0 ]; then
> break
> fi
> echo "Square of $num: $((num ** 2))"
> done
Enter a number: 5
Square of 5: 25
Enter a number: 8
Square of 8: 64
Enter a number: -11

Admin@DESKTOP-OEUSPON ~
$
```

Part E

Q.1.

(1) Consider a 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 the average waiting time using FCFS scheduling.

Process	Arrival time	Burst time	response time	waiting time.
P_1	0	5	0	0
P_2	1	3	5	4
P_3	2	6	8	6

Gant chart:

P_1	P_2	P_3	
0	5	8	14

Waiting time = CT − Arrival time − Burst time

P_2 = 8 − 1 − 3
= 7 − 3
= 4

Average waiting time = $\frac{\text{total no. of waiting time}}{\text{~~time~~ total no. of process}}$

$= \frac{10}{3}$

$= 3.33.$

Q.2.

(2) Consider the following processes with arrival time and burst time.
Calculate the average turn around time using shortest job first scheduling.

By using shortest job First scheduling;

Process	Arrival time	Burst time	completion time	Turnaround time.
P_3	2	1	3	1
P_1	0	3	6	6
P_2	1	5	11	10
P_4	3	4	15	12

the shortest burst time is executed first;

$\therefore$ P_3 ---- burst time = 1
P_1 ---- burst time = 3
P_4 ---- burst time = 4
P_2 ---- burst time = 5.

Turnaround time = completion time − Arrival time.

$\therefore$ $P_3 = (3-2) = 1$
$P_1 = (6-0) = 6$
$P_2 = (11-1) = 10$
$P_4 = (15-3) = 12$

Average turnaround time $= (1+6+10+12)/4$

$= \frac{29}{4} = 7.25$

$\therefore$ Avg. Turnaround time $= 7.25$.

3.

③ Consider the following processes with arrival time, burst time, and priorities (lower number indicates higher priority):
Calculate the average waiting time using priority scheduling.

→ In priority scheduling, the process with the highest priority (lowest number) is executed first.

Process	Arrival time	Burst time	Priority	waiting time	Turnaround Time.
P_2	1	4	1	0	4
P_4	3	2	2	4	6
P_1	0	6	3	6	12
P_3	2	7	4	12	19

$P_2 \Rightarrow$ Priority 1
$P_4 \Rightarrow$ Priority 2
$P_1 \Rightarrow$ Priority 3
$P_3 \Rightarrow$ Priority 4

$\therefore$ WT = Turnaround Time − Burst time.

∴ & for priority scheduling:

WT = Sum of Burst times of all previous processes with higher priority.

∴ P_2 ; WT = 0 .... its the first process to execute.

P_4 , WT = 4 .... it takes 4 units to complete.

P_1 , WT = 6 .... it takes 4 + 2 = 6 units to complete.

P_3 , WT = 12 .... it takes 4 + 2 + 6 = 12 units to complete.

∴ Average waiting = (0 + 4 + 6 + 12) / 4

$$= \frac{22}{4} = 5.5$$

Average waiting time = $\underline{\underline{5.5}}$

Concepts of Operating System- Assignment 2 March 1, 2025

Name: Shital Kawthe_DAC_KH

Q.4.

④ Consider the following processes with arrival times and burst times, and the time quantum for Round robin scheduling is 2 unit:

calculate the average turnaround time Round Robin scheduling.

Process	Arrival time	Burst time	completion time	Turnaround time.
P_1	0	4	6	6
P_2	1	5	11	10
P_3	2	2	4	2
P_4	3	3	9	6

in round robin scheduling, each process is executed for fixed time quantum.

∴ In this case, 2 units.

- P_1 (0 – 2)
- P_2 (2 – 3)
- P_3 (3 – 4)
- P_1 (4 – 6)
- P_2 (6 – 8)
- P_4 (8 – 9)
- P_2 (9 – 11)

Average Turnaround time $= (6+10+2+6)/4$

$= \frac{24}{4} = 6.$

$\therefore$ Ave. turnaround time $= 6.$

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?

After the fork() call, both the parent and child processes will have their own separate copies of the variable x. This is because fork() creates a new process by duplicating the existing process, including its memory space.

Initially, the parent process has x = 5.

After forking:

- The parent process still has x = 5 and increments it to x = 6.

- The child process inherits a copy of x with the value x = 5 and increments it to x = 6.

So, after the fork() call, both the parent and child processes will have x = 6.

Here's a simple illustration:

Parent Process (before fork):

x = 5

After fork():

Parent Process:

x = 5 → x = 6 (after increment)

Child Process:

x = 5 (inherited) → x = 6 (after increment)

Note that since the child process has its own separate copy of x, changes made by the child process do not affect the parent process's variable x, and vice versa.