

CDAC MUMBAI

Concepts of Operating System

Assignment 2

PART A

Question. What will the following commands do?

1. echo "Hello, World!"

It will print the string Hello World!

2. name="Productive"

It will assign string (Productive) to variable (name) in the shell script.

3. touch file.txt

It will create an empty file with the name of "file.txt".

4. ls -a

It will list all the files and directories which are present in the current directory along with the hidden files.

5. rm file.txt

It will remove the file from current directory.

6. cp file1.txt file2.txt

This command will copy the details of file1.txt into file2.txt

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

This command will move the file (file.txt) to the path (/path/to/directory/).

8. chmod 755 script.sh

The above command is used to change the mode of permissions for user, group, and owner of the bash script (script.sh). It means that the user has all the permissions for reading, writing, and executing the file but the group and owner have the permissions for reading and executing it.

9. grep "pattern" file.txt

This command is used to match the word "pattern" in the file (file.txt).

10. kill PID

The command sends a termination signal to the specific ID of process.

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

It will create a directory (mydir) and then change the directory to mydir. Then, create an empty file (file.txt), print “Hello, World!” in the file.txt and then display the “Hello, World!”.

The output will be: Hello, World!

12. ls -l | grep ".txt"

This command lists the detailed information of files like their permissions, owner, userID, etc. Along with it, it will also display the text files which matches with the “.txt”.

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

It will display the information of file1.txt and file2.txt. Then, it passes the output of cat to the next command and sort it alphabetically without duplicate consecutive lines using “uniq” command.

14. ls -l | grep "^d"

The “ls -l” command lists all the detailed information of all files and directories. The pipe function is used to pass the output of “ls -l” command to “grep “^d”” for displaying only those directories whose lines start with “d”.

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

The “grep” command searches for a specific “pattern” in all the files and sub-directories inside /path/to/directory/.

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

cat command displays the information of file1.txt and file2.txt. Then, the pipe function passes this output to the sort command and it sorts the combined lines alphabetically and then it will display only duplicate lines.

17. chmod 644 file.txt

The chmod command changes the permissions of file.txt in which the owner has permission for reading and write. User and others have the permission to read only.

18. cp -r source_directory destination_directory

This command copies all the files and directories of source_directory into destination_directory.

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

It searches for those files ending with .txt in the “/path/to/search” directory.

20. chmod u+x file.txt

It changes the permission of file.txt and gives permission of executing the file to owner (user).

21. echo \$PATH

It displays the environment variables that stores those directories where executable files are present.

PART B

Question. 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.

False

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.

False

Question. 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.

```
echo "Hello, World!"
```

```
cdac@DESKTOP-7ULKR5D:~$ nano main.sh1
cdac@DESKTOP-7ULKR5D:~$ bash main.sh1
Hello, World!
```

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

```
name="CDAC Mumbai"
```

```
echo $name
```

```
cdac@DESKTOP-7ULKR5D:~$ nano print_variable.sh1
cdac@DESKTOP-7ULKR5D:~$ bash print_variable.sh1
CDAC Mumbai
cdac@DESKTOP-7ULKR5D:~$ |
```

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

```
echo "Enter a number: "
```

```
read number
```

```
echo "Entered Number is: $number"
```

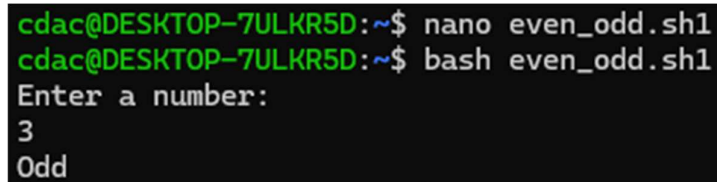
```
cdac@DESKTOP-7ULKR5D:~$ nano input_number.sh1
cdac@DESKTOP-7ULKR5D:~$ bash input_number.sh1
Enter a number:
3
Entered Number is: 3
```

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

```
cdac@DESKTOP-7ULKR5D:~$ nano addition.sh1
cdac@DESKTOP-7ULKR5D:~$ bash addition.sh1
The sum of 5 and 3 is: 8
```

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

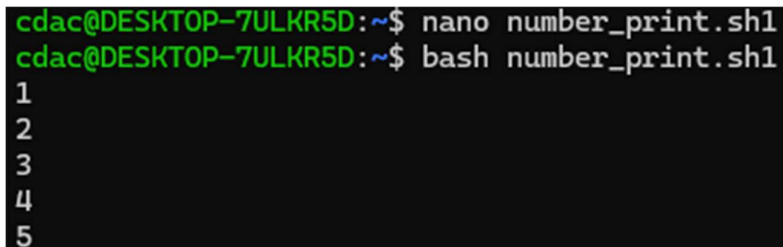
```
echo "Enter a number: "  
  
read num  
  
if ((num % 2 == 0)); then  
  
    echo "Even"  
  
else  
  
    echo "Odd"  
  
fi
```



```
cdac@DESKTOP-7ULKR5D:~$ nano even_odd.sh1  
cdac@DESKTOP-7ULKR5D:~$ bash even_odd.sh1  
Enter a number:  
3  
Odd
```

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
```



```
cdac@DESKTOP-7ULKR5D:~$ nano number_print.sh1  
cdac@DESKTOP-7ULKR5D:~$ bash number_print.sh1  
1  
2  
3  
4  
5
```

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

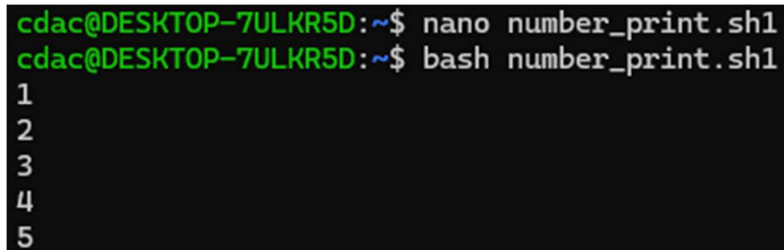
```
num=1  
  
while [ $num -le 5 ]  
  
do
```



```
    echo $num

    ((num++))

done
```



```
cdac@DESKTOP-7ULKR5D:~$ nano number_print.sh1
cdac@DESKTOP-7ULKR5D:~$ bash number_print.sh1
1
2
3
4
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".

```
file="file.txt"

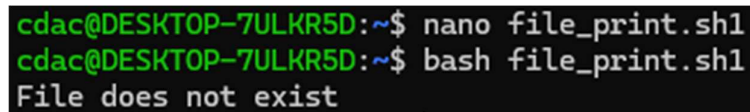
if [ -f "$file" ]; then

    echo "File exists"

else

    echo "File does not exist"

fi
```



```
cdac@DESKTOP-7ULKR5D:~$ nano file_print.sh1
cdac@DESKTOP-7ULKR5D:~$ bash file_print.sh1
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.

```
echo "Enter a number: "

read num

if [ $num -gt 10 ]; then

    echo "The number is greater than 10."

else

    echo "The number is 10 or less."
```

fi

```
cdac@DESKTOP-7ULKR5D:~$ nano check_number.sh1
cdac@DESKTOP-7ULKR5D:~$ bash check_number.sh1
Enter a number:
4
The number is 10 or less.
```

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.

```
for i in {1..5}
```

```
do
```

```
    for j in {1..5}
```

```
    do
```

```
        echo $((i * j))
```

```
    done
```

```
    echo
```

```
done
```

```
cdac@DESKTOP-7ULKR5D:~$ bash table.sh1
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
```

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
```

```
    echo "Enter a number (negative to exit):"
```

```
    read num
```

```
    if [ $num -lt 0 ]; then
```

```
        echo "Negative number entered."
```

```
        break

    fi

    square=$((num * num))

    echo "Square of $num is: $square"

done
```

```
cdac@DESKTOP-7ULKR5D:~$ bash square_number.sh1
Enter a number (negative to exit):
3
Square of 3 is: 9
Enter a number (negative to exit):
4
Square of 4 is: 16
Enter a number (negative to exit):
5
Square of 5 is: 25
Enter a number (negative to exit):
-4
Negative number 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.

Answer

Process	Arrival Time	Burst Time	Completion time	Turn Around Time	Waiting Time
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6

Gantt Chart

P0	P1	P2	P3
0	5	8	14

Turn Around Time = Completion Time – Arrival Time

Waiting Time = Turn Around Time – Burst Time

Average Waiting Time = $(0+4+6)/3 = 10/3 = 3.33$

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.

Answer

Gantt Chart

P1	P3	P4	P2	
0	3	4	8	13

Process	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time (TAT) = CT - AT
P1	0	3	3	3
P3	2	1	4	2
P4	3	4	8	5
P2	1	5	13	12

Average TAT = $(3+2+5+12)/4$

$$= 22/4 = 5.5$$

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.

Answer

Gantt Chart:

P1	P2	P4	P3
0	6	10	12

Process	Arrival Time	Burst Time	Completion Time (CT)	TAT	WT
P1	0	6	6	6	0
P2	6	4	10	9	5
P4	10	2	12	9	7
P3	12	7	19	17	10

Average WT = $0+5+7+10/4$

$$= 22/4 = 5.5$$

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.

Answer

P1	P2	P3	P4	P1	P2	P4	P2
0	2	4	6	8	10	12	13

Process	Arrival Time	Completion Time (CT)	TAT
P1	0	8	8

P2	1	13	12
P4	2	6	4
P3	3	12	9

Average TAT = $33/4 = 8.2$

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 fork():

x = 5 (only the parent exists at this point).

After fork():

Two separate processes now exist, each with its own copy of x = 5.

Child Process: Increments x \rightarrow x = 6.

Parent Process: Increments x \rightarrow x = 6.

Final Values of x:

Parent Process: x = 6

Child Process: x = 6