CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  - It prints the Hello, World. Whichever text is available in double quotes.
- name="Productive"
  - It creates a name variable and assign the Productive value to it.
- touch file.txt
  - It create a new file called file.txt
- ls -a
  - It shows all file in current directory including hidden file
- rm file.txt
  - It removes or delete the file.txt file.
- cp file1.txt file2.txt
  - it copy the file1.txt into file2.txt
- mv file.txt /path/to/directory/
  - It move the file.txt to another directory. Like from source to destination.
- chmod 755 script.sh
  - it will change the permission of user, group and owner of the file, 755 represent in binary. In that we can give a permission of read, write and execute.
- grep "pattern" file.txt
  - It finds the matching words in the file and give only the matches words.
- kill PID
  - It terminates the process using process ID.
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  - It creates the mydir(directory) then go in mydir directory, then we create a file.txt and in that file we write an helloworld script. After that we have printed that hello world using cat command.

- ls -l | grep ".txt"
  - it will find in current directory which matches the .txt files.

- cat file1.txt file2.txt | sort | uniq
  - it combines the file1 and file2.txt file and sort them and removes the duplicates to diplay only unique content

- ls -l | grep "^d"
  - It lists files in details and filters only the entries that are directory.

- grep -r "pattern" /path/to/directory/
  - it searches the text pattern in all file in the another directory.

- cat file1.txt file2.txt | sort | uniq –d
  - it combines the file1 and file2.txt file and sort them and show unique and only duplicate lines.

- chmod 644 file.txt

  - It sets the permissions of file.txt so the owner can read & write, while the group and others can only read.

- cp -r source_directory destination_directory
  - it copys the entire source directory into destination directory

- find /path/to/search -name "*.txt"
  - It searches inside /path/to/search for files whose names end with .txt and lists their paths.

- chmod u+x file.txt
  - it giving the execute permission to the owner.

- echo $PATH

  It displays the value of the PATH environment variable, which is a list of directories the shell searches to find executable programs.

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**
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. **- chmod**
2. **cpy** is used to copy files and directories. **- cp**
3. **mkfile** is used to create a new file. **- touch**
4. **catx** is used to concatenate files. **- cat**
5. **rn** is used to rename files. **- mv**

Part C

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

```
#!/bin/bash
echo "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
```

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 the result.

```
a=5
b=3
sum=$((a + b))
echo "The sum of $a and $b is: $sum"
```

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
```

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 $i
  i=$((i+1))
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".

```
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.

```
echo "Enter a number:"
read num

if [ $num -gt 10 ]; then
  echo "The number is greater than 10"
else
  echo "The number is not greater than 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.

```
for i in {1..5}
do
  for j in {1..5}
  do
    printf "%4d" $((i * j))
  done
  echo
done
```

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 number to exit):"
    read num

    if [ $num -lt 0 ]; then
        echo "Negative number entered. Exiting..."
        break
    fi

    square=$((num * num))
    echo "Square of $num is: $square"
done
```

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.

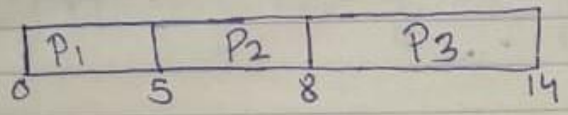
5. Consider a program that uses the **fork()** system call to create a child process. Initially, the parent process has a variable **x** with a value of 5. After forking, both the parent and child processes increment the value of **x** by 1.
   What will be the final values of **x** in the parent and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

Q 1.

~~Process Arrival time Burst time~~

Process	Arrival Time	Burst Time	completion Time	TAT	WT
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6
					3.33

Gantt chart

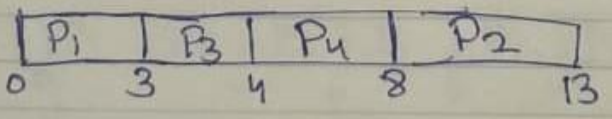


$$\text{Avg WT} = \frac{10}{3} = \underline{\underline{3.33}}$$

Q2.

Process	Arrival Time	Burst Time	Completion Time	TAT
P1	0	3	3	3
P2	1	5	13	12
P3	2	1	4	2
P4	3	4	8	5

Gantt chart



$$\text{Avg TAT} = \frac{22}{4} = \underline{\underline{5.5}}$$

Q3.

Process	Arrival Time	Burst Time	Priority	CT	TAT	WT
✓ P1	0	6	3	6	6	0
P2	1	4	1	10	9	5
P3	2	7	4	19	17	10
P4	3	2	2	12	9	7
						5.5

Gantt chart

P1	P2	P4	P3

0 6 10 12 19

$$\text{Avg WT} = \frac{22}{4} = 5.5$$

Q4. TQ = 2 unit.

Process	Arrival Time	Burst Time	CT	TAT	WT
P1	0	~~4~~ ~~2~~ 0	8	8	4
P2	1	~~5~~ ~~3~~ 1	14	13	8
✓ P3	2	2	~~13~~6	4	2
P4	3	~~3~~ 1	13	10	7

Gantt Chart

P1	P2	P3	P1	P4	P2	P4	P2

0 2 4 6 8 10 12 13 14

Ready queue.

~~P1~~	~~P2~~	~~P3~~	~~P1~~	~~P4~~	~~P2~~	P4	P2

Avg TAT = 8.75 Avg WT = 5.25