CDAC MUMBAI

Concepts of Operating System
Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  > It will print Hello, World!
- name="Productive"
  > It will create a variable named name and assign value Productive
- touch file.txt
  > It will create an empty file named file.txt
- ls -a
  > It will list all files as well as hidden files in the current directory
- rm file.txt
  > It will delete the file file.txt
- cp file1.txt file2.txt
  > It will copy the content of file1.txt into file2.txt
- mv file.txt /path/to/directory/
  > It will move the file named file.txt to the specified directory
- chmod 755 script.sh
  > It will set the permissions of script.sh for owner, group and others.
  Owner can read, write and execute
  Group can read and execute
  Others can read and execute
- grep "pattern" file.txt
  > It will search for lines having “pattern” word in file.txt and prints them.
- kill PID
  > It will terminate the process with PID (Process ID).
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  > It will make a directory named mydir, then moves into the directory, creates a file named file.txt, write “Hello, World!” into it and then displays the file content.
- ls -l | grep ".txt"
  > It will list all the files with detailed information and filter only files containing .txt.
- cat file1.txt file2.txt | sort | uniq
  > It will concatenate file1.txt and file2.txt, then sorts the lines and removes the duplicate lines.
- ls -l | grep "^d"
  > It will list only directories in the current folder.

- grep -r "pattern" /path/to/directory/
  > It will search for “pattern” in all files in the given directory as well as files inside the subdirectories.
- cat file1.txt file2.txt | sort | uniq –d
  > It will show only duplicate lines in file1.txt and file2.txt files.
- chmod 644 file.txt
  > It will set permissions for file.txt
  Owner can read and write
  Group can read
  Others can read
- cp -r source_directory destination_directory
  > It will copy the source directory and all its subdirectories and files into the destination directory.
- find /path/to/search -name "*.txt"
  > It will search for files ending with .txt under the specified path.
- chmod u+x file.txt
  > It will add execute permission for the owner of file.txt
- echo $PATH
  > It will display the current PATH environment variable, which lists directories the shell 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 copy files and directories.
   > False. cd command is used to change directories, not copy. Copy files and directories is done with cp command.
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.
   > The command is Incorrect, the correct command to change file permissions is chmod.
2. cpy is used to copy files and directories.
   > The command is Incorrect, the correct command to copy files and directories is cp.
3. mkfile is used to create a new file.
   > The command is Incorrect for Linux System, mkfile exists in other systems like Solaris, the correct command to create a new file in Linux systems is touch command.
4. catx is used to concatenate files.
   > The command is Incorrect, the correct command to concatenate files is cat.
5. rn is used to rename files.
   > The command is Incorrect, the correct command to rename files is mv.

Part C

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

```
#!/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.
Code:

```
#!/bin/bash
name="CDAC Mumbai"
echo $name
```

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

```
#!/bin/bash
read -p "Enter a number: " 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.

Code:

```
#!/bin/bash
a=5
b=3
sum=$((a + b))
echo "Sum: $sum"
```

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

Code:

```
#!/bin/bash
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.

Code:

```
#!/bin/bash
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.

Code:

```
#!/bin/bash
i=1
while (( i <= 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".

Code:

```
#!/bin/bash
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.

Code:

```
#!/bin/bash
read -p "Enter a number: " num
if (( num > 10 ))
then
   echo "Number is greater than 10"
else
   echo "Number is 10 or less"
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.

Code:

```
#!/bin/bash
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.

Code:

```
#!/bin/bash
while true
do
   read -p "Enter a number: " num
   if (( num < 0 ))
   then
      echo "Negative number entered. Exiting the Loop."
      break
   fi
   echo "Square: $((num * num))"
```

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.

Solution:

Process	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6

Average Waiting time (AWT):

AWT = (0+4+6)/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.

Solution:

Process	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P1	0	3	3	3	0
P2	1	5	13	12	7
P3	2	1	4	2	1
P4	3	4	8	5	1

Average Turnaround Time (TAT):

TAT = (3+2+5+12)/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.

Solution:

Process	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P1	0	6	3	6	0
P2	1	4	1	9	5
P3	2	7	4	9	7
P4	3	2	2	17	10

Average Waiting Time (AWT):

AWT = (0+5+7+10)/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.

Solution:

Process	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P1	0	4	10	10	6
P2	1	5	14	13	8
P3	2	2	6	4	2
P4	3	3	13	10	7

Average Turnaround Time (TAT):

ATAT = (10+13+4+10)/4 = 9.25

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?

Solution:
fork() is used to create a new child process in the Linux Operating System.
The child process gets a copy of the parent's memory like stack, heap, variables, etc.

Before fork() the value of variable x in the parent process is 5.
Parent process: x = 5

After fork() a child process is created which will get a copy of the parent's memory.
Parent process: x = 5
Child process: x = 5

Now both parent and child processes are incrementing the value of x by 1.

Parent process: x = 5 + 1 = 6
Child process: x = 5 + 1 = 6

So, the final values of x in the parent and child processes after the fork call:
Parent: x = 6
Child: x = 6