CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  = It will print the string as it is.

- name="Productive"
  =assigns the string "Productive" to a variable named name. This variable is stored in the current shell session and can be accessed or used later in scripts or commands.

- touch file.txt
  =create file in current directory named file.txt.

- ls -a
  =show all directories and files even hidden files in that current directory where (.) represent current directory and (..) represent parent directory

- rm file.txt
  =remove file from the directory.

- cp file1.txt file2.txt
  =Copy content of file.txt to file2.txt.

- mv file.txt /path/to/directory/
  = moves the file file.txt to the specified directory /path/to/directory/

- chmod 755 script.sh
  = chmod Changes file permissions and 755 Sets specific permission levels: Owner=**7** (Read, Write, Execute), Group → 5 (Read, Execute), Others → 5 (Read, Execute) . script.sh is a Target file for which permissions are set.

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

- kill PID
  =Terminate the process of particular process Id (PID)

- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  =
  mkdir mydir= will create directory
  &&= this operators ensure thet each command runs only if previous one successfully done.
  cd mydir= It will open mydir directory.
  touch file.txt= It will create file.txt named file.
  echo "Hello, World!" > file.txt = echo will print “Hello World” and > this sign will transfer that “Hello World” to file.txt
  cat file.txt = will print “Hello World”

- ls -l | grep ".txt"
  = Lists files and directories in the current directory in long format (showing permissions, owner, size, and modification time). Sends the output of ls -l as input to the grep command. Searches for lines containing .txt in the output of ls -l.

- cat file1.txt file2.txt | sort | uniq
  =It will sort content of file1.txt and file2.txt in an alphabetical order and align in one pipe line and after that it will identify unique names or numbers in both the files.

- ls -l | grep "^d"
  = It will show only directories which are present in ls -1 .

- grep -r "pattern" /path/to/directory/
  =Searches “pattern” recursively through all subdirectories inside /path/to/directory/

- cat file1.txt file2.txt | sort | uniq –d
  shows only duplicate lines and avoid unique ones in the content of file1.txt and file2.txt after sorting those files in one pipeline in an alphabetical order.

- chmod 644 file.txt
  =Changes the file permissions as owner can read and write , group can read , Others can also read.

- cp -r source_directory destination_directory
  =Copy source directory recursively into destination directory.

- find /path/to/search -name "*.txt"
  =This searches for .txt into the path /to/search -name

- chmod u+x file.txt
  =The file will become executable for owner

- echo $PATH
  = It displays the current directories in the PATH environment variable, which determines where the system looks for executables when you run a command.

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

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 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 number
  You have Entered=$number

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

```
- echo “Enter two numbers for addition”
  read num1
  read num2
  add=$(($num1 + $num2))
  echo “You have entered two numbers $num1 and $num2, Addition of both the numbers is $add”
```

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 number
if((number % 2 == 0));
then
{
echo "Even"
}
fi
```

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

```
- a=0
for a in 1 2 3 4 5
do
echo $a
done
```

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

```
- a=1
while [ $a -lt 6 ]
do
echo $a
a=`expr $a + 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 number
if(($number > 10));
then
echo "Number is greater than10"
else
echo "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.

```
- echo "Multiplication table from 1 to 5"
for i in 1 2 3 4 5
do
for j in 1 2 3 4 5
do
result=$((i*j))
printf "%4d" " $result"
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 -n "Enter a number (negative number to exit): "
    read number
    if (( number < 0 )); then
        echo "Negative number entered. Exiting..."
        break
    fi
    square=$((number * number))
    echo "Square of $number 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.

PID	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	5	0	0	5
P2	1	3	5	4	7
P3	2	6	8	6	12

Average Waiting Time= (0+4+6)/3 = 3.34

Gantt Chart=

P1	P2	P3		
0 5	5 8	8 14		

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.

PID	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	3	1	2
P4	3	4	4	1	5

Gantt chart

Average Turnaround Time= (3+12+2+5)/ 4 =5.5

P1	P1	P1	P3	P4	P2	
0 - 1	1 - 2	2 - 3	3 - 4	4 - 8	8 - 13	

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.

PID	Arrival Time	Burst Time	Priority	Response Time	Waiting Time	Turnaround Time
P1	0	6	3	0	6	12
P2	1	4	1	1	0	4
P3	2	7	4	12	10	17
P4	3	2	2	5	2	4

Average waiting Time=
(6+0+10+2)/4 – 4.5

Gantt Chart-

P1	P2	P4	P1	P3		
0 1	1 5	5 7	7 12	12 19		

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.

PID	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	4	0	6	10
P2	1	5	2	9	14
P3	2	2	4	2	4
P4	3	3	6	5	8

Average Turnaround Time=
(10+14+4+8)/4 - 9

Gantt Chart-

P1	P2	P3	P4	P1	P2	P4	P2							
0-2	2-4	4-6	6-8	8-10	10-12	12-13	13-14							

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?

Answer=
Parent class- x=6
Child class- x=6

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.