CDAC MUMBAI

Concepts of Operating System

Assignment 2

What will the following commands do?

1. echo "Hello, World!"

Prints "Hello, World!" to the console.

2. name="Productive"

Assigns the string "Productive" to the variable 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 files.

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 string "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, changes 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 then 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 then removes duplicate lines.

14. ls -l | grep "^d"

Lists all files and directories in the current directory in a detailed format, and then searches for directories (lines starting with d).

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

Searches for the string "pattern" recursively in all files within the specified directory.

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

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

Copies the entire source_directory to the destination_directory, including all subdirectories and files.

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 contains the list of directories to search 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

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. **False**

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.
   Incorrect. The correct command is chmod.

2. **cpy** is used to copy files and directories.
   Incorrect. The correct command is cp.

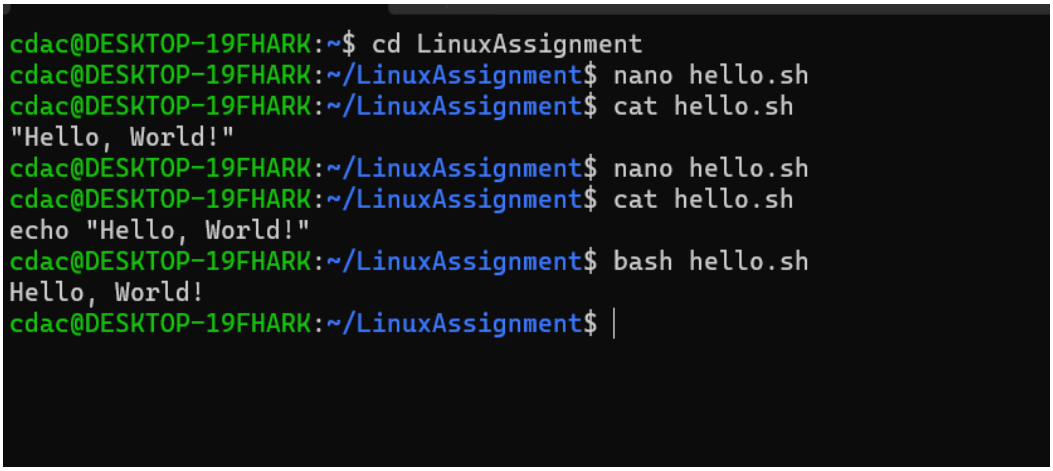
3. **mkfile** is used to create a new file.
   Incorrect. The correct command is touch.

4. **catx** is used to concatenate files.
   Incorrect. The correct command is cat.

5. **rn** is used to rename files.
   Incorrect. The correct command is mv or rename.

Part C

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



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

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano name.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash name.sh
CDAC Mumbai
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q3.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q3.sh
echo "Enter a number"
read a
echo Your number is $a

cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q3.sh
Enter a number
98
Your number is 98
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q4.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q4.sh
echo "Enter a number"
read a
echo "Enter a number"
read b
sum=`expr $a + $b`
echo sum of $a and $b is $sum
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q4.sh
Enter a number
5
Enter a number
3
sum of 5 and 3 is 8
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q5.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q5.sh
echo Enter number to check even or odd:
read num
if [ $((num % 2)) -eq 0 ]
then
echo The $num is even.
else
echo The $num is odd.
fi
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q5.sh
Enter number to check even or odd:
5
The 5 is odd.
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q5.sh
Enter number to check even or odd:
4
The 4 is even.
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

Q6. Write a shell script that uses a for loop to print numbers from 1 to 5.

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q6.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q6.sh
no=1
for no in {1..5}
do
echo $no
done
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash.sh
bash.sh: command not found
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q6.sh
1
2
3
4
5
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

Q7. Write a shell script that uses a while loop to print numbers from 1 to 5.

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q7.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q7.sh
no=1
for no in {1..5}
do
echo $no
done
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q7.sh
1
2
3
4
5
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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".

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q8.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q8.sh
file="file.txt"
if [ -e "$file" ]
then
        echo "File Exists"
else
        echo "File don't exist"
fi
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q8.sh
File don't exist
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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.

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q9.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q9.sh
echo Enter any number:
read no
if [ $no -gt 10 ]
then
echo The $no is greater than 10.
else
echo The $no is not greater than 10.
fi
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q9.sh
Enter any number:
11
The 11 is greater than 10.
cdac@DESKTOP-19FHARK:~/LinuxAssignment$
```

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.

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q10.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q10.sh
for i in {1..5}
do
for j in {1..10}
do
echo $i "*" $j = $(( i * j ))
done
echo
done
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q10.sh
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
```

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.

```
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ nano Q11.sh
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ cat Q11.sh
echo Enter numbers"(to stop loop enter negative number)":
while true
do
read no
if [ $no -lt 0 ]
then
echo entered negative number..exiting..
break
fi
square=expr $((no*no))
echo Square of $no is $square
echo Want to exit enter negative number..
done
cdac@DESKTOP-19FHARK:~/LinuxAssignment$ bash Q11.sh
Enter numbers(to stop loop enter negative number):
5
Q11.sh: line 10: 25: command not found
Square of 5 is
Want to exit enter negative number..
```

Part E

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?

Ans:

After the fork() call:

Parent Process:
1. Initial value of x: 5
2. Increment x by 1: x = 5 + 1 = 6

Child Process:
1. Initial value of x: 5 (copied from parent process)
2. Increment x by 1: x = 5 + 1 = 6

Both the parent and child processes will have a final value of x = 6.

1. What is an operating system & what are its primary functions?

- An operating system is system software that manages computer hardware & software resources.
- Acts as an interface between the user & hardware

primary functions : process management
memory management
file system management
Device management
Security & acess control
User interface management

2. Explain the difference betn process & thread.

- process : An independent program in execution with its own memory & resources.
- Thread - A lightweight unit of a process that shares the process's memory & resources

3. What is Virtual memory and How does it work?

- A memory management technique that provides an illusion of a larger RAM
- Uses a portion of disk storage to extend RAM

Working : - The os moves data between RAM and disk as needed
- Uses paging or segmentation to manage memory
- Allows execution of large programs that exceed physical memory

4. Describe the difference betn multiprogramming, multitasking & multiprocessing.

→ - multiprogramming: multiple programs in memory, cpu switches betn them (no time-sharing)

- multitasking: cpu switches rapidly between tasks, giving the illusion of parallel execution

- multiprocessing: Multiple cpus or cores executing different processes simultaneously.

5. What is a file system & what are its components?

- A file system is a method of storing and organizing data on storage devices.
- It's components include files, directories, inodes, metadata & file allocations structures

6. What is a deadlock & how can it be prevented?

- A deadlock occurs when processes wait indefinitely for resources held by each other
- prevention techniques:
  - Resources ordering (assign a fixed order for resource allocation)
  - Avoid hold & wait (request all resources at once)
  - preemption (forcefully take resources from a process)
  - Avoid circular wait (ensure no cyclic dependencies)

7. Explain the difference betn a kernel & a shell.
- Kernel: The core of the os managing hardware and system processes.

- shell: The user interface that allows interaction with the os (CLI or GUI)

8. What is cpu scheduling and why is it important?
- cpu scheduling is the process of selecting which process will execute next to ensure efficient cpu utilization.
- It's important for responsiveness, fairness & process execution efficiency.

9. How does a system call work?
- A system call allows user programs to request services from the operating system.
- It involves switching from user mode to kernel mode, executing the request operation & returning the result.

10. What is the purpose of device drivers in an operating system?
- Device drivers act as intermediaries betn Hardware device & the operating system, allowing communication & control over peripherals

11. Explain the role of the page table in virtual memory management?

- stores the mapping betⁿ logical/virtual addresses & physical addresses
- Helps in translating virtual memory addresses to physical memory addresses.
- Used in demand paging to track memory pages and their locations.

12. What is thrashing and how can it be avoided

- Thrashing occurs when excessive paging/swappi reduces system performance.
- causes: Too many processes competing for limited memory.

prevention Techniques:

- Increase RAM
- use better page replacement algorithm e.g. LRU
- Reduce degree of multiprogramming

13. Describe the concept of a semaphore and its use in synchronization.

- A semaphore is a synchronization mechanism used to manage access to shared resources.

Types:

Binary semaphore (0 or 1)
counting semaphore (values greater than 1)

Use:

prevents race conditions by allowing only a limited number of processes to access a resource.

14. How does an operating system handle process synchronization?

→ - It uses mechanisms like semaphores, mutex & monitors to ensure that multiple processes or threads do not ~~share~~ interfere each other

15. What is the purpose of an interrupt in operating system?

- ~~Interrupts~~ Interrupts notify the cpu about an event that needs immediate attention.

Types:
- Hardware interrupts (e.g. keyboard input, disk I/O)
- Software interrupts (e.g. system calls, exceptions).

16. Explain the concept of a file descriptor.

- A unique integer assigned to an open file in a process.
- Used in systems calls for reading, writing & managing files.

17. How does a system recover from a system crash?

Recovery mechanism:
- checkpointing: Save system state periodically
- Backup and restore: Load previous stable ~~stat~~ state
- Journaling file system: Log all file system changes before committing.

Q6 What is process control block (PCB) and what information does it contain?
- A data structures that stores process related information
contains:- process ID
- program counter
- cpu register
- Memory limits
- process state

Q7 Describe the process state diagram and transitions beth different process states
States:
- New: process created
- Ready: waiting for cpu
- Running: currently executing
- Waiting: waiting for I/O
- Terminated: Execution Finished

Q8 How does a process communicate with another process in an operating system?
→ Methods:
- Inter-process communication (IPC) mechanism
- shared memory
- Message passing (pipes, sockets)

Q9. What is process synchronisation and why is it important?
- Ensures multiple processes execute in order without conflicts.
- prevents: Race conditions, data inconsistency

Q30. Explain the concept of a zombie process and How it is created
- A process that has completed execution but still has entry in process table
- created when a parent does not read the exit status of a child process

31. What is demand paging and How does it improve memory management efficiency?
- Loads only required pages into memory, reducing memory usage
- Uses page faults to load missing pages from disk

32. How does a memory management unit (MMU) Works
- Translates virtual addresses into physical addresses
- Uses a page table to track memory mappings

33. What is system call & How does it facilitate communication betn User programs and the OS
- System calls allow programs to request OS service (e.g. file operations, process control)
e.g. open(), read(), write(), Fork(), exec()

34. Explain the concept of a race condition and how it can be prevented.
- occurs when multiple processes access shared resources simulatanously.
prevention -
use locks, semaphores, mutexes
Ensure atomic operations

35. What is a zombie process and how does it occur? How can a zombie process be prevented?
→ occurs when a child process terminates but parent does not call wait().
prevention: parent should call wait() or use signal handling.

36. Explain the concept of an orphan process. How does an operating system handle orphan processes?
- A process whose parent has terminated
- OS assign it to the process (PID 1) to manage it

37. How does the fork() system call work in creating a new process in Unix-like operating systems?
- creates a new child process that is a duplicate of the parent.
- Both continue execution from the next instruction

38. Describe how a parent process can wait for a child process to finish execution.

→ - Uses the wait() system call
- prevents zombie processes by collecting exit status.

39. How can a parent process terminate a child process in Unix-like operating systems?

- Using the kill() system call with the child's PID

40. What is the purpose of the waitpaid() system call in process management? How does it differ from wait()?

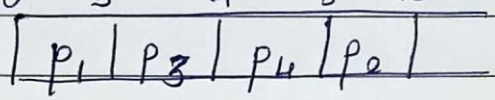
- Waitpid() waits for a specific child process
- wait() waits for any child process

Q2. Consider the following process with arrival times and burst times.

process	Arrival T	Burst T	Response T	Waiting T	TAT
P_1	0	3	0	0	~~12~~ 3
P_2	1	5	8	7	12
P_3	2	1	3	1	2
P_4	3	4	4	1	5

calculate the average turn around time using shortest job first (SJF) scheduling.

→ Gault chart



$$\text{Avg Turnaround time} = \frac{3+12+2+5}{4}$$

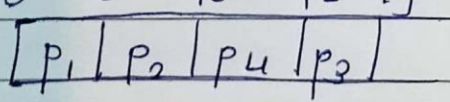
$$= \frac{22}{4} = 5.5$$

Q3. consider the following processes with arrival time, burst times and priorities (lower number indicates higher priority.

process	Arrival T	Burst T	priority	waiting time
P_1	0	6	3	0
P_2	1	4	1	5
P_3	2	7	4	10
P_4	3	2	2	7

calculate the avg. waiting time using priority scheduling

→ Gault chart -



$$\text{Avg. waiting Time} - \frac{0+5+10+7}{4} = \frac{22}{4} = 5.5$$

Part E

Q1. Consider the following processes with arrival times and burst times

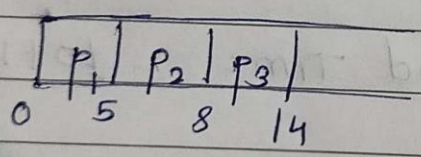
process	Arrival time	Burst Time
P_1	0	5
P_2	1	3
P_3	2	6

Calculate the average waiting time using first-come first served (FCFS) scheduling

→

process	Arrival time	BurstT	ResponseT	WaitingTAT
P_1	0	5	0	0-0 = 0
P_2	1	3	5	5-1 = 4
P_3	2	6	8	8-2 = 6

Gantt chart:



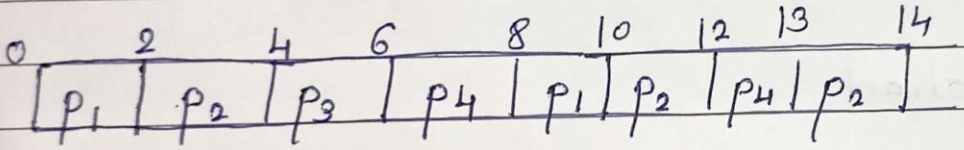
∴ Average waiting time = $\frac{0+4+6}{3}$

$= \frac{10}{3} = 3.3$

Q4 Consider the following processes with arrival times and burst times and the time quantum for Round Robin scheduling is 2 units.

process	Arrival T	Burst T.	waiting time	Turn around T
P_1	0	4	6	10
P_2	1	5	8	13
P_3	2	2	2	4
P_4	3	3	7	10

Gault chart:



Avg. Turnaround Time $= \frac{10+13+4+10}{4}$

$= \frac{37}{4}$

$= 9.25$