CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

Submitted by : Aman Garg_KH

What will the following commands do?

• echo "Hello, World!"

This command is used to print Hello, World! On the terminal.

```
cdac@DESKTOP-4G80EEJ:~$ echo "Hello, World!"
Hello, World!
```

• name="Productive"

This command is used to create a variable named name which stores value as Productive.

```
cdac@DESKTOP-4G80EEJ:~$ name="Productive"
cdac@DESKTOP-4G80EEJ:~$ echo $name
Productive
```

• touch file.txt

This command is used to create a text file named file.txt.

```
cdac@DESKTOP-4G80EEJ:~$ touch file.txt
cdac@DESKTOP-4G80EEJ:~$ cat file.txt
cdac@DESKTOP-4G80EEJ:~$ ls
LinuxAssignment  file.txt       question11.sh       q
```

• ls -a

This command lists all the directories and files in the current directory including the hidden files also.

```
cdac@DESKTOP-4G80EEJ:~$ ls -a
.              .bashrc      .profile                 file.txt           question3.sh  question7.sh
..             .cache       .sudo_as_admin_successful  question10.sh      question4.sh  question8.sh
.bash_history  .local       LinuxAssignment          question11.sh      question5.sh  question9.sh
.bash_logout   .motd_shown  Program.c                question11.sh.save  question6.sh
cdac@DESKTOP-4G80EEJ:~$
```

• rm file.txt

This command is used to delete the file file.txt.

```
cdac@DESKTOP-4G80EEJ:~$ rm file.txt
cdac@DESKTOP-4G80EEJ:~$ ls
LinuxAssignment  question10.sh  question11.sh.save  question4.sh  question6.sh  question8.sh
Program.c        question11.sh  question3.sh        question5.sh  question7.sh  question9.sh
```

• cp file1.txt file2.txt

This command copies the content of file1.txt to file2.txt.

```
cdac@DESKTOP-4G80EEJ:~$ nano file1.txt
cdac@DESKTOP-4G80EEJ:~$ cp file1.txt file2.txt
cdac@DESKTOP-4G80EEJ:~$ cat file1.txt
hello there
cdac@DESKTOP-4G80EEJ:~$ cat file2.txt
hello there
cdac@DESKTOP-4G80EEJ:~$
```

• mv file.txt /path/to/directory/

This command is used to move the file named file.txt to this path "/path/to/directory/" and if the file is in the same directory this can be used to rename it.

```
cdac@DESKTOP-4G80EEJ:~$ mv file.txt ~/LinuxAssignment/
```

```
cdac@DESKTOP-4G80EEJ:~$ cd LinuxAssignment
cdac@DESKTOP-4G80EEJ:~/LinuxAssignment$ ls
data.txt  docs.zip       file.txt   fruit.txt  newdirectory  output.txt
docs      duplicate.txt  file1.txt  input.txt  numbers.txt
```

• chmod 755 script.sh

This command grants permission to script.sh which means the owner can now read, write and execute the file.

```
cdac@DESKTOP-4G80EEJ:~$ touch script.sh
cdac@DESKTOP-4G80EEJ:~$ chmod 755 script.sh
```

```
-rwxr-xr-x 1 cdac cdac    0 Aug 30 16:38 script.sh
```

• grep "pattern" file.txt

This command is used to find the certain type of word in the file. Like here the word "pattern" is searched in file.txt and if it exists it is printed on the terminal.

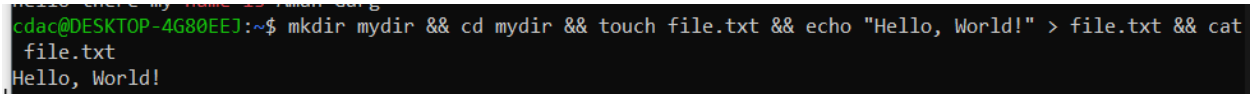
```
cdac@DESKTOP-4G80EEJ:~$ nano file.txt
cdac@DESKTOP-4G80EEJ:~$ grep "name is" file.txt
hello there my name is Aman Garg
```

• kill PID

This command is used to terminate the process with specifying process id (PID).

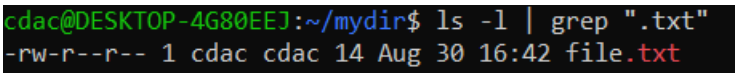
• mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

This command creates a directory named mydir then it changes the current directory to mydir and then a empty file named file.txt is created under mydir. Then echo "Hello, World!" > file.txt: writes “Hello, World!” to file.txt. Then content of file.txt is shown onto the terminal.



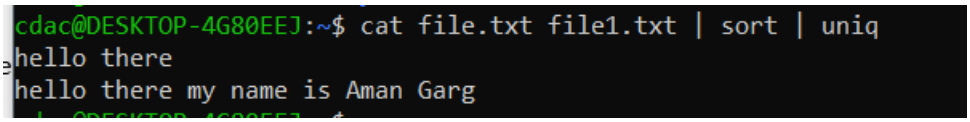
• ls -l | grep ".txt"

This command lists files in long format and pipes the output to grep, which filters and displays only the lines containing .txt .



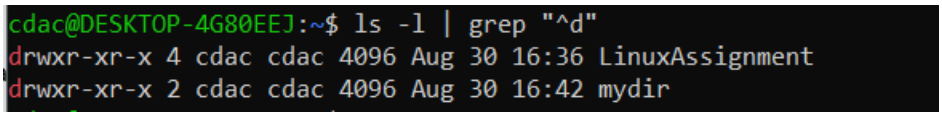
• cat file1.txt file2.txt | sort | uniq

This command concatenates file1.txt and file2.txt , then sorts the combined output and removes the duplicates.



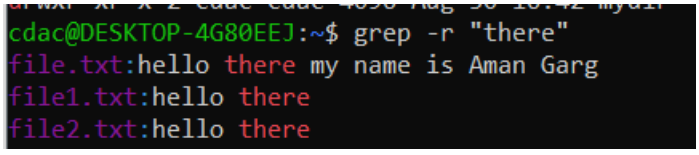
• ls -l | grep "^d"

This command lists files in long format (ls -l) and filters to show only directories.



• grep -r "pattern" /path/to/directory/

This command recursively searches for the string "pattern" in all files within /path/to/directory/ and its subdirectories.



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

This command concatenates file1.txt and file2.txt , then sorts the combined output and then displays only the duplicate lines.

• chmod 644 file.txt

This command changes the permissions of file.txt to 644, which means the owner can read and write, while others can only read.

```
-rw-r--r-- 1 cdac cdac   33 Aug 30 16:41 file.txt
```

• cp -r source_directory destination_directory

This command recursively copies the “source directory” to “destination directory”.

• find /path/to/search -name "*.txt"

This command searches for all files ending in .txt within /path/to/search and its subdirectories.

```
cdac@DESKTOP-4G80EEJ:~$ find -name "*.txt"
./file.txt
./file1.txt
./mydir/file.txt
./file2.txt
./LinuxAssignment/fruit.txt
./LinuxAssignment/numbers.txt
./LinuxAssignment/file.txt
./LinuxAssignment/input.txt
./LinuxAssignment/data.txt
./LinuxAssignment/newdirectory/docs/file2.txt
./LinuxAssignment/duplicate.txt
./LinuxAssignment/file1.txt
./LinuxAssignment/docs/file2.txt
./LinuxAssignment/output.txt
```

• chmod u+x file.txt

This command gives permission to user with execute on file.txt.

```
-rwxr--r-- 1 cdac cdac   33 Aug 30 16:41 file.txt
```

• echo $PATH

Prints the current value of the path and lists directories where shell looks for executable files.

```
cdac@DESKTOP-4G80EEJ:~/mydir$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/
mnt/c/Program Files/WindowsApps/CanonicalGroupLimited.Ubuntu22.04LTS_2204.3.63.0_x64__79rhkp1fndgsc:/mnt/c/
Program Files/Common Files/Oracle/Java/javapath:/mnt/c/Windows/system32:/mnt/c/Windows:/mnt/c/Windows/Syste
m32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0/:/mnt/c/Windows/System32/OpenSSH/:/mnt/c/Program Fi
les/Java/jdk-17.0.1/bin:/mnt/c/MinGW/bin:/mnt/c/Program Files/nodejs/:/mnt/c/Program Files/Git/cmd:/mnt/c/P
rogram Files/MySQL/MySQL Shell 8.0/bin/:/mnt/c/Users/amang/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users
/amang/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/amang/AppData/Roaming/npm:/mnt/c/Users/ama
ng/AppData/Local/GitHubDesktop/bin:/mnt/c/Users/amang/AppData/Local/Programs/mongosh/:/snap/bin
```

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.
   Correct command → chmod is used to change file permissions.

2. **cpy** is used to copy files and directories.
   Correct command → cp is ued to copy files and directories.

3. **mkfile** is used to create a new file.
   Correct command → touch is used to create a new file.

4. catx is used to concatenate files.
   Correct command → cat is used to concatenate files.

5. rn is used to rename files.
   Correct command → mv is used to rename files.

Part C

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

```
cdac@DESKTOP-4G80EEJ:~$ echo "Hello, World!"
Hello, World!
```

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

```
cdac@DESKTOP-4G80EEJ:~$ name="CDAC Mumbai"
cdac@DESKTOP-4G80EEJ:~$ echo $name
CDAC Mumbai
```

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

```
cdac@DESKTOP-4G80EEJ:~$ nano question3.sh
cdac@DESKTOP-4G80EEJ:~$ bash question3.sh
Enter number:
10
10
```

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

```
var1=3
var2=5
sum=$(($var1 + $var2))
echo "the sum is : $sum"
```

```
cdac@DESKTOP-4G80EEJ:~$ nano question4.sh
cdac@DESKTOP-4G80EEJ:~$ bash question4.sh
the sum 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 the number:"
read number1
rem=$(( $number1 % 2 ))
if [ $rem -eq 0 ];
then
        echo "Entered number is even"
else
        echo "Entered number is odd"
fi
```

```
cdac@DESKTOP-4G80EEJ:~$ nano question5.sh
cdac@DESKTOP-4G80EEJ:~$ bash question5.sh
Enter the number:
2
Entered number is even
```

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

```
 GNU nano 6.2                          question6.sh
echo "Enter the number to which you want to print:"
read num
for ((i=1; i<=5; i++))
do
        echo "Number: $i"
done
```

```
cdac@DESKTOP-4G80EEJ:~$ nano question6.sh
cdac@DESKTOP-4G80EEJ:~$ bash question6.sh
Enter the number to which you want to print:
5
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

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

```
 GNU nano 6.2                          question7.sh
read count = 1
while [ $count -le 5 ]
do
        echo $count
        ((count++))
done
```

```
cdac@DESKTOP-4G80EEJ:~$ nano question7.sh
cdac@DESKTOP-4G80EEJ:~$ bash question7.sh
1
question7.sh: line 1: read: `=': not a valid identifier
1
2
3
4
5
cdac@DESKTOP-4G80EEJ:~$
```

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".

```
 GNU nano 6.2                                                question8.sh
filename="file.txt"
if [ -f "$filename" ]
then
        echo "File exists"
else
        echo "File doesnt exists"
fi
```

```
cdac@DESKTOP-4G80EEJ:~$ nano question8.sh
cdac@DESKTOP-4G80EEJ:~$ bash question8.sh
File exists
cdac@DESKTOP-4G80EEJ:~$
```

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-4G80EEJ:~$ nano question8.sh
cdac@DESKTOP-4G80EEJ:~$ nano question9.sh
cdac@DESKTOP-4G80EEJ:~$ bash question9.sh
enter the number:
9
The number is less than 10
cdac@DESKTOP-4G80EEJ:~$ bash question9.sh
enter the number:
11
The number you entered is greater than 10
cdac@DESKTOP-4G80EEJ:~$
```

```
 GNU nano 6.2                                                question9.sh
echo "enter the number:"
read num1

if [ $num1 -gt 10 ]
then
        echo "The number you entered is greater than 10"
else
        echo "The number is less 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.

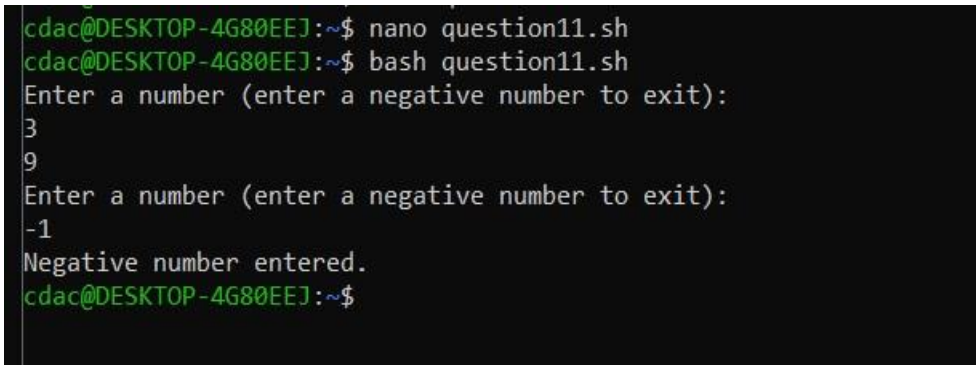
```
cdac@DESKTOP-4G80EEJ:~$ nano question10.sh
cdac@DESKTOP-4G80EEJ:~$ bash question10.sh
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
cdac@DESKTOP-4G80EEJ:~$
```

```
 GNU nano 6.2                                        question10.sh
for i in {1..5}
do
        for j in {1..10}
        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.

```
 GNU nano 6.2                                        question11.
while true
do
        echo "Enter a number (enter a negative number to exit):"
        read num1
        if [ $num1 -lt 0 ]
then
        echo "Negative number entered."
        break
        fi

        result=$(($num1*$num1))
        echo $result
done
```



Part D

Common Interview Questions (Must know)

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

Ans. Operating system is software that manages computer hardware and software resources and provides services for computer programs. Primary functions include managing hardware resources (CPU, memory, I/O devices), providing a user interface, executing and managing tasks, and handling system security and file management.

2. Explain the difference between process and thread.

Ans. Processes are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution. PCB (Process Control Block) holds the context of process. A process can create other processes which are known as Child Processes. The process takes more time to terminate, and it is isolated means it does not share the memory with any other process. The process can have the following states new, ready, running, waiting, terminated, and suspended.

Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process. A thread has three states: Running, Ready, and Blocked.

The thread takes less time to terminate as compared to the process but unlike the process, threads do not isolate.

3. What is virtual memory, and how does it work?

Ans. A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory.

4. Describe the difference between multiprogramming, multitasking, and multiprocessing.

Ans. Multiprogramming: Multiple programs are loaded into memory and executed by the CPU one by one.

Multitasking: Multiple tasks or processes are executed simultaneously by rapidly switching between them.

Multiprocessing: Multiple CPUs or cores process multiple tasks simultaneously.

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

Ans. A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device.

Components include the boot sector, file allocation table (FAT), directories, and files.

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

Ans. A deadlock is a situation where a set of processes is stuck in a state where each process is waiting for a resource held by another, leading to an indefinite halt. It can be prevented by ensuring at least one of the following conditions does not hold: mutual exclusion, hold and wait, no preemption, and circular wait.

7. Explain the difference between a kernel and a shell.

Ans. The kernel is the core part of the OS that interacts with hardware and manages resources. The shell is a user interface that provides access to the services of the OS, often through command-line input.

8. What is CPU scheduling, and why is it important?

Ans. The kernel is the core part of the OS that interacts with hardware and manages resources. The shell is a user interface that provides access to the services of the OS, often through command-line input**.**

9. How does a system call work?

Ans. A system call is a way for a program to request services from an operating system's kernel. This process allows programs to access privileged functionalities that the operating system provides.

10. What is the purpose of device drivers in an operating system?

Ans. Device Driver in computing refers to a special kind of software program or a specific type of software application that controls a specific hardware device that enables different hardware devices to communicate with the computer's Operating System.

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

Ans. The page table maps virtual addresses to physical addresses. It keeps track of where the OS has stored data in physical memory (or on disk) corresponding to the virtual memory used by processes.

12. What is thrashing, and how can it be avoided?

Ans. Thrashing occurs when a system spends more time swapping pages in and out of memory than executing processes. It can be avoided by using better page replacement algorithms.

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

Ans. Semaphores are used to manage concurrent processes by controlling access to shared resources. Semaphores can signal and wait, ensuring that critical sections are accessed by only one process at a time.

14. How does an operating system handle process synchronization?

Ans. OSs use synchronization mechanisms like mutexes, semaphores, and monitors to coordinate processes, ensuring that shared data is accessed in a controlled and safe manner, preventing race conditions and data inconsistency.

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

Ans. An interrupt is a signal that causes an operating system (OS) to stop a current process or service and take a specific action. Interrupts can be generated by a device attached to the computer or by a program running on the computer.

16. Explain the concept of a file descriptor.

Ans. A file descriptor is an abstract identifier used by a process to access a file or other input/output resource, such as a socket. It's a handle through which a process interacts with these resources.

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

Ans. After a system crash, the OS may use techniques like checkpointing, logs, and backup copies to restore the system to a stable state. Recovery processes include restarting services, restoring data from backups, and repairing corrupted files.

18. Describe the difference between a monolithic kernel and a microkernel.

Ans. The kernel manages the operations of the computer, In microkernel, the user services and kernel services are implemented in different address spaces. The user services are kept in the user address space, and kernel services are kept under the kernel address space.

In a Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services are implemented in the same address space.

19. What is the difference between internal and external fragmentation?

Ans. Internal fragmentation: Wasted space within allocated memory blocks due to allocation units being larger than the requested memory.

External fragmentation: Free memory is scattered, making it difficult to allocate contiguous blocks despite sufficient total free space.

20. How does an operating system manage I/O operations?

Ans.. The OS manages the I/O operations by using interrupts, buffering data, device drivers.

21. Explain the difference between preemptive and non-preemptive scheduling.

Ans. Preemptive scheduling: The OS can interrupt a running process to assign the CPU to another process.

Non-preemptive scheduling: A running process must finish its CPU burst before the CPU can be assigned to another process.

22. What is round-robin scheduling, and how does it work?

Ans. Round-robin scheduling is a preemptive scheduling algorithm where each process is assigned a fixed time slice (quantum). Processes are cycled through in a queue, ensuring each gets a fair share of CPU time.

23. Describe the priority scheduling algorithm. How is priority assigned to processes?

Ans. In priority scheduling, processes are assigned priorities, and the CPU is allocated to the process with the highest priority.

Priorities can be assigned based on factors like process type, resource needs, or user-defined criteria.

24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?

Ans. SJN (or SJF, Shortest Job First) selects the process with the smallest execution time to run next. It's used to minimize waiting time and optimize overall CPU usage, but it can lead to starvation for longer processes.

25. Explain the concept of multilevel queue scheduling.

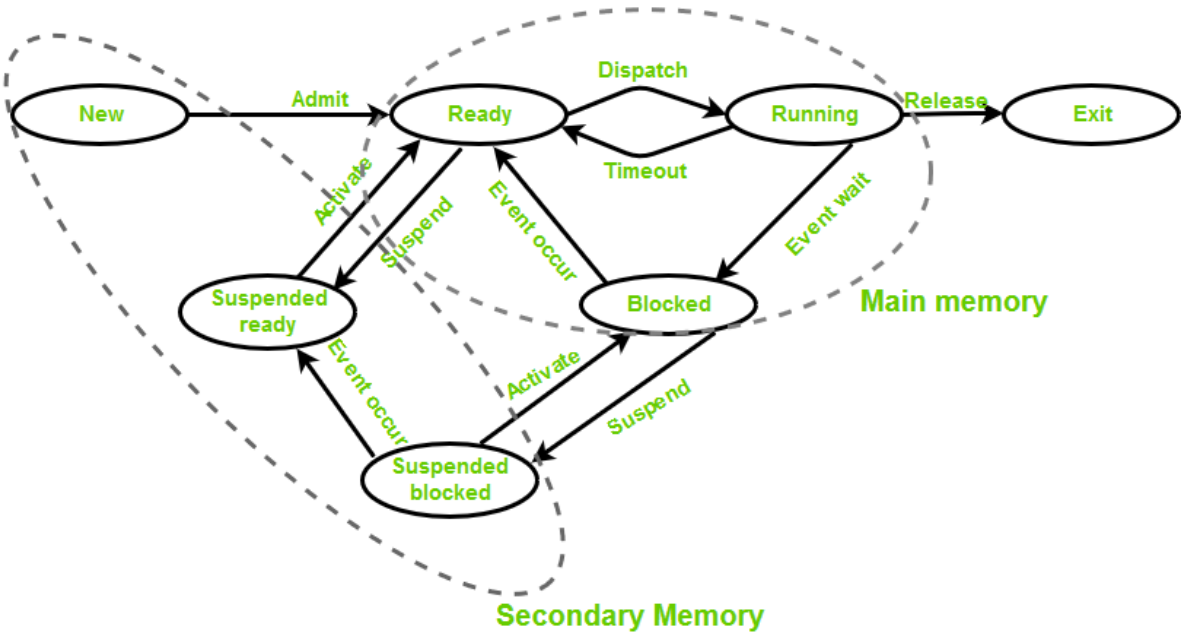
Ans. Multilevel queue scheduling divides the ready queue into several separate queues, each with its scheduling algorithm.

26. What is a process control block (PCB), and what information does it contain?

Ans. The PCB is a data structure in the OS that stores all the information about a process, including its process ID, state, CPU registers, scheduling information, memory management information, and I/O status.

27. Describe the process state diagram and the transitions between different process states.

Ans. A process typically transitions through the following states: New (created), Ready (waiting for CPU), Running (executing), Waiting (blocked, waiting for an event), Terminated (finished). Transitions occur due to events like process creation, I/O completion, or CPU preemption.



28. How does a process communicate with another process in an operating system?

Ans. Processes can coordinate and interact with one another using a method called inter-process communication (IPC) .

29. What is process synchronization, and why is it important?

Ans. Process synchronization ensures that concurrent processes operate correctly when accessing shared resources, preventing race conditions and ensuring data integrity.

30. Explain the concept of a zombie process and how it is created.

Ans. A zombie process occurs when a child process has terminated, but its exit status has not been read by its parent process. It remains in the process table until the parent reads the status or terminates.

31. Describe the difference between internal fragmentation and external fragmentation.

Ans. Internal fragmentation: Wasted space within allocated memory blocks due to allocation units being larger than the requested memory.

External fragmentation: Free memory is scattered, making it difficult to allocate contiguous blocks despite sufficient total free space.

32. What is demand paging, and how does it improve memory management efficiency?

Ans. Demand paging loads pages into memory only when they are needed, reducing the memory usage and improving efficiency by avoiding loading unnecessary pages.

33. Explain the role of the page table in virtual memory management.

Ans. The page table maps virtual addresses to physical addresses. It keeps track of where the OS has stored data in physical memory corresponding to the virtual memory used by processes.

34. How does a memory management unit (MMU) work?

Ans. MMU stands for Memory management unit also known as PMMU (paged memory management unit), Every computer system has a memory management unit , it is a hardware component whose main purpose is to convert virtual addresses created by the CPU into physical addresses in the computer’s memory. In simple words, it is responsible for memory management.

35. What is thrashing, and how can it be avoided in virtual memory systems?

Ans. Thrashing occurs when a system spends more time swapping pages in and out of memory than executing processes, usually due to insufficient memory. It can be avoided by adjusting the degree of multiprogramming or using better page replacement algorithms.

36. What is a system call, and how does it facilitate communication between user programs and the operating system?

Ans. A system call provides a controlled entry point to the services offered by the OS. When a user program requests a service from the OS, it triggers a system call, which switches the process to kernel mode and executes the corresponding kernel function.

37. Describe the difference between a monolithic kernel and a microkernel.

Ans. The kernel manages the operations of the computer, In microkernel, the user services and kernel services are implemented in different address spaces. The user services are kept in the user address space, and kernel services are kept under the kernel address space.

In a Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services are implemented in the same address space.

38. How does an operating system handle I/O operations?

Ans. The OS manages the I/O operations by using interrupts, buffering data, device drivers.

39. Explain the concept of a race condition and how it can be prevented.

Ans. A race condition occurs when the outcome of a process depends on the timing of other processes, leading to unpredictable results. It can be prevented using synchronization mechanisms like locks, semaphores.

40. Describe the role of device drivers in an operating system.

Ans. Device drivers are specialized programs that allow the OS to communicate with hardware devices.

41. What is a zombie process, and how does it occur? How can a zombie process be prevented?

Ans. A zombie process occurs when a child process has terminated, but its exit status has not been read by its parent process. It remains in the process table until the parent reads the status or terminates.

A zombie process can be prevented by ensuring the parent process calls wait() or waitpid() to read the child's exit status.

42. Explain the concept of an orphan process. How does an operating system handle orphan processes?

Ans. An orphan process occurs when its parent process terminates before the child process. The OS handles orphan processes by reassigning them to the init process (PID 1), which then waits for their termination.

43. What is the relationship between a parent process and a child process in the context of process management?

Ans. A parent process creates a child process, which inherits most of the parent's attributes, including environment variables, file descriptors, and memory segments. The parent can control or communicate with the child process.

44. How does the fork() system call work in creating a new process in Unix-like operating systems?

Ans. In many operating systems, the fork system call is an essential operation. The fork system call allows the creation of a new process. When a process calls the fork(), it duplicates itself, resulting in two processes running at the same time. The new process that is created is called a child process. It is a copy of the parent process. The fork system call is required for process creation and enables many important features such as parallel processing, multitasking, and the creation of complex process hierarchies.

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

Ans. A parent process can use the wait() or waitpid() system calls to pause execution until one or more child processes have terminated, allowing the parent to retrieve the child's exit status.

46. What is the significance of the exit status of a child process in the wait() system call?

Ans. The exit status returned by wait() indicates whether the child process terminated normally or abnormally.

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

Ans. A parent process can terminate a child process using the kill() system call.

48. Explain the difference between a process group and a session in Unix-like operating systems.

Ans. A process group is a collection of related processes which can all be signalled at once.

A session is a collection of process groups, which are either attached to a single terminal device (known as the controlling terminal) or not attached to any terminal.

49. Describe how the exec() family of functions is used to replace the current process image with a new one.

Ans. The exec() family of functions replaces the current process image with a new program image, effectively running a new program within the same process. The process ID remains the same, but the code, data, and stack are replaced.

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

Ans. waitpid() allows a parent to wait for a specific child process to terminate, identified by its PID. Unlike wait(), which waits for any child process, waitpid() can be more selective and can also wait for non-child processes if used with certain flags.

51. How does process termination occur in Unix-like operating systems?

Ans. Process termination occurs when a process finishes executing or is killed by a signal.

52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?

Ans. The long-term scheduler controls the admission of processes into the system, determining which processes should be brought into the ready queue. It influences the degree of multiprogramming by controlling the number of processes loaded into memory.

53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?

Ans. The short-term scheduler (CPU scheduler) makes frequent, quick decisions about which process should run next, usually based on a scheduling algorithm. The long-term scheduler runs less frequently, deciding which processes should enter the ready queue. The medium-term scheduler swaps processes in and out of memory to balance load and manage memory.

54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

Ans. The medium-term scheduler may be invoked when the system is low on memory, and processes need to be swapped out to free up space. It helps by reducing the degree of

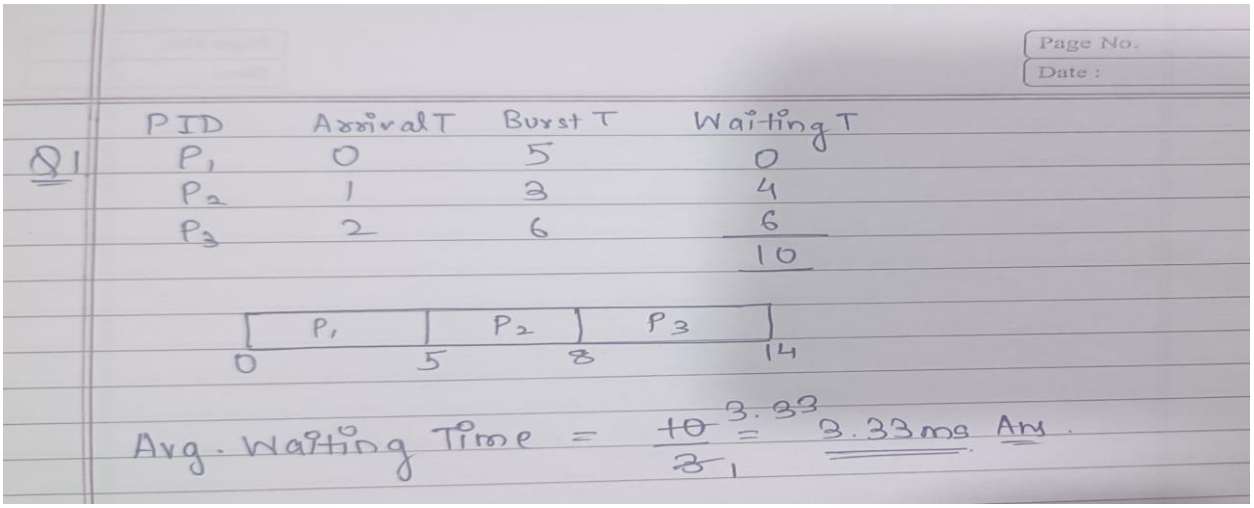
multiprogram ming temporarily, ensuring the most important processes have enough resources to run efficiently.

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

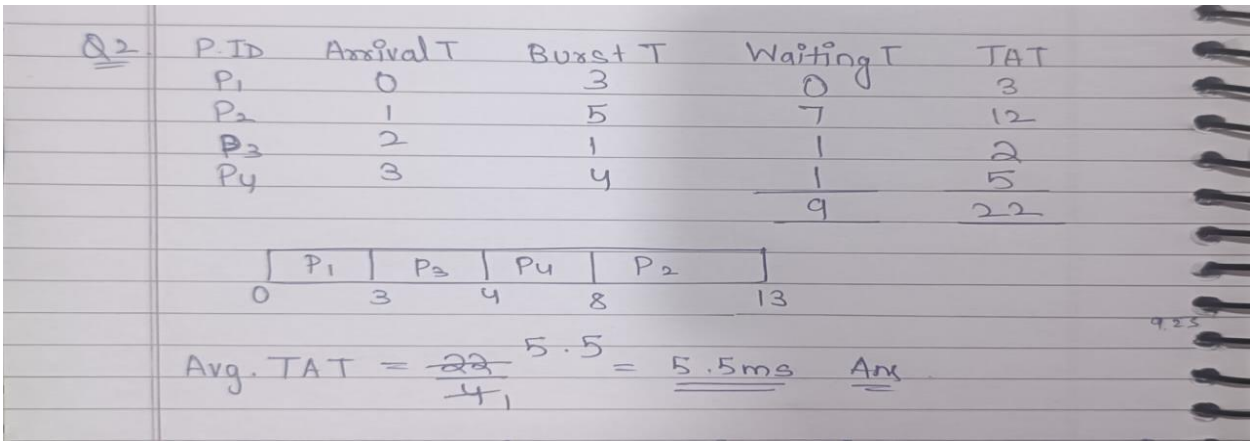
Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.



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

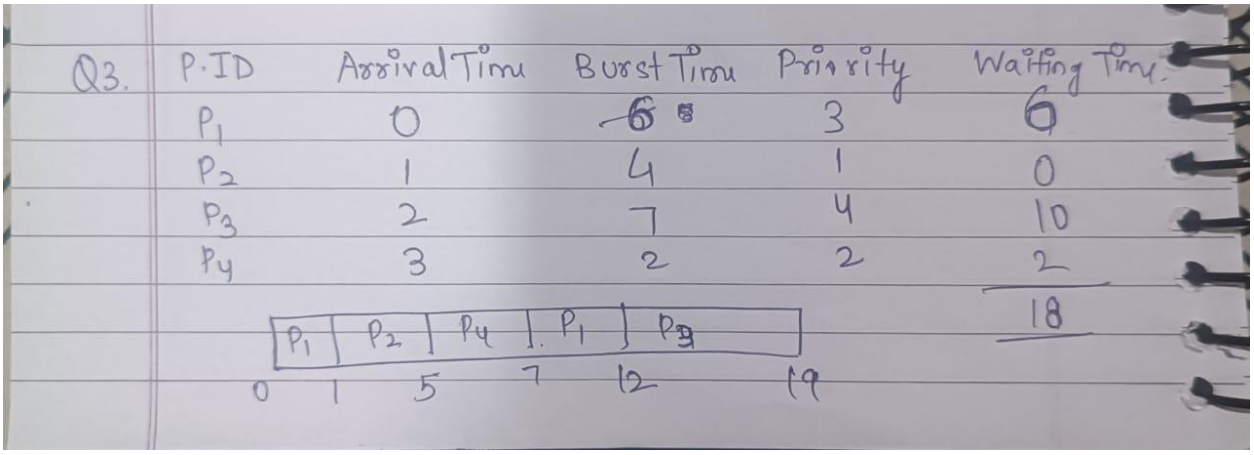
Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

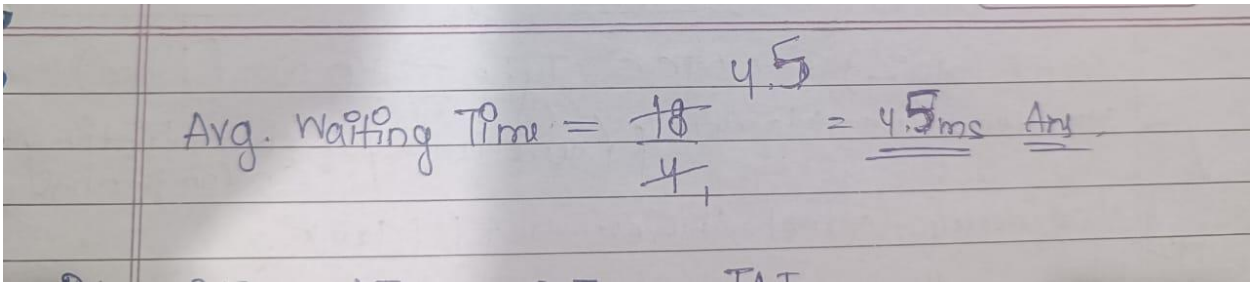


3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

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:

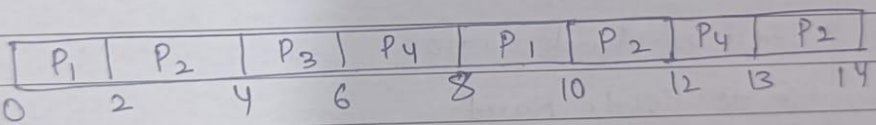
Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

Q4

P.ID	AT	BT	TAT
P_1	0	4	10
P_2	1	5	13
P_3	2	2	4
P_4	3	3	10
			37

Time Quantum = 2 units



Avg. TAT = 37/4 = 9.25 ms Ans

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?

Q5Ans. Initial value of x : 5

After fork() system call creates a child process. The child process is duplicate of parent process. Now, Both parent & child processes have their own copy of x with the initial value of 5.

So, if x is incremented by 1 then, the value will be updated in both parent & child process variable.

Final Value :- Parent process → x : 6
Child process → x : 6