**Nishant Patil**

**PG-DAC**

**Concept of Operating Systems**

# **Part A**

***What will the following commands do?***

* **echo "Hello, World!"**

**:-** It will print the "Hello, World!" sentence in terminal.

* **name="Productive"**

:- It will store “Productive” in ‘name’ variable.

* **touch file.txt**

:- It will create file named file.txt

* **ls -a**

**:-** List out all the files including hidden ones.

* **rm file.txt**

:- Remove the file named file.txt from directory.

* **cp file1.txt file2.txt**

**:-** Copying contents of file1.txt into file2.txt

* **mv file.txt /path/to/directory/**

:- Moving file.txt to different directory.

* **chmod 755 script.sh**

**:-** changing the permissions of script.sh to read,write and execute by owner; read and execute by group; read and execute by others.

* **grep "pattern" file.txt**

**:-** Finding “pattern” inside the file.txt

* **kill PID**

:- Terminating process of given ID.

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

:- Creating new directory mydir, changing directory to mydir, creating new file.txt, appending “Hello, World!” in file.txt and showing the contents of file.txt i.e "Hello, World!"

* **ls -l | grep ".txt"**

:- Listing out the contents in the directory with their permissions and finding “.txt” in it.

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

**:-** Showing contents in file1.txt and file2.txt, sorting those contents and displaying unique content.

* **ls -l | grep "^d"**

**:-** To list directories within a given directory.

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

**:-** Searching "pattern" in given directorie in recursive manner.

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

:- Displaying duplicated contents in file1.txt and file2.txt only once in sorted manner.

* **chmod 644 file.txt**

**:-** Changing the permissions of file.txt to read and write by owner; read by group; read by others**.**

* **cp -r source\_directory destination\_directory**

**:-**  Copying contents of source directory to destination directory in recursive manner.

* **find /path/to/search -name "\*.txt"**

**:-** Searching contents with “.txt” in it in directory which’s path given.

* **chmod u+x file.txt**

**:-** Giving user permission of execute the file.txt.

* **echo $PATH**

:- Printing the value of the $path environment variable.

# **Part B**

***Identify True or False:***

1. **ls is used to list files and directories in a directory.**

**:- True**

1. **mv is used to move files and directories.**

**:- True**

1. **cd is used to copy files and directories.**

**:- False**

1. **pwd stands for "print working directory" and displays the current directory.**

**:- True**

1. **grep is used to search for patterns in files.**

**:- True**

1. **chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.**

**:- True**

1. **mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.**

**:- True**

1. **rm -rf file.txt deletes a file forcefully without confirmation.**

**:- True**

***Identify the Incorrect Commands:***

1. **chmodx is used to change file permissions.**

**:- Incorrect** ( chmod is used to change file permissions. )

1. **cpy is used to copy files and directories.**

**:- Incorrect** ( cp is used to copy files and directories. )

1. **mkfile is used to create a new file.**

**:- Incorrect** (touch is used to create a new file. )

1. **catx is used to concatenate files.**

**:- Incorrect** (cat is used to concatenate files. )

1. **rn is used to rename files.**

**:- Incorrect**( mv is used to rename files.)

# **Part C**

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

:-

echo "Hello, World!"

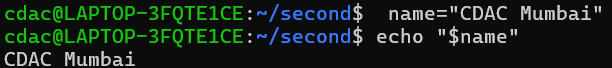


***Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.***

:-

name="CDAC Mumbai"

echo $name



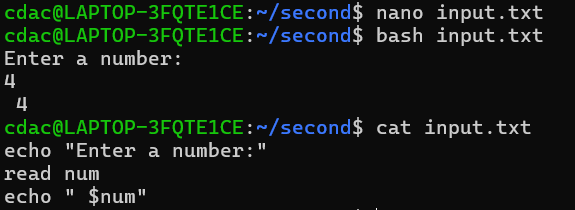
***Question 3: Write a shell script that takes a number as input from the user and prints it.***

:-

echo "Enter a number:"

read num

echo " $num"



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

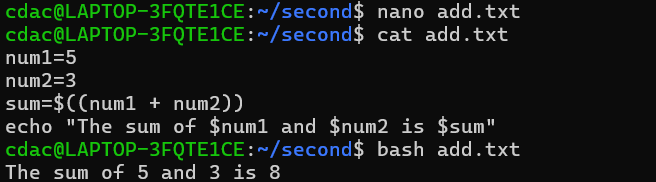
:-

num1=5

num2=3

sum=$((num1 + num2))

echo "The sum of $num1 and $num2 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 number

if [$((number % 2)) -eq 0]

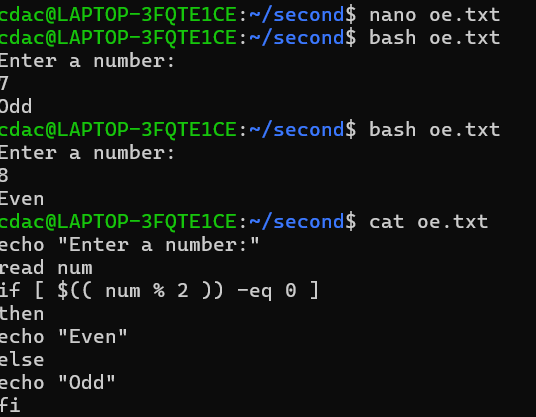
then

echo "Even"

else

echo "Odd"

fi



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

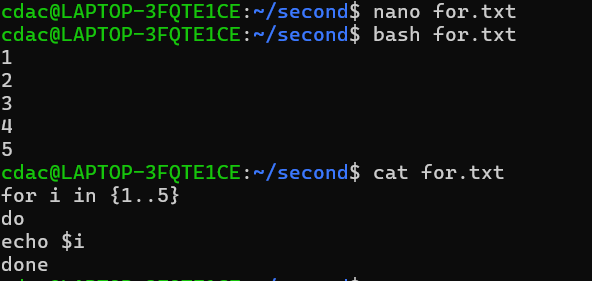
:-

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

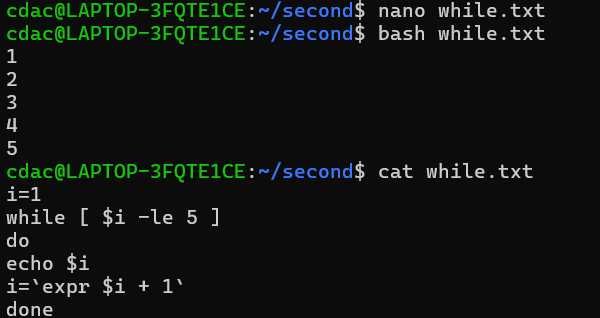
while [ $i -le 5 ]

do

echo $i

i=`expr $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 [ -e "file.txt" ]

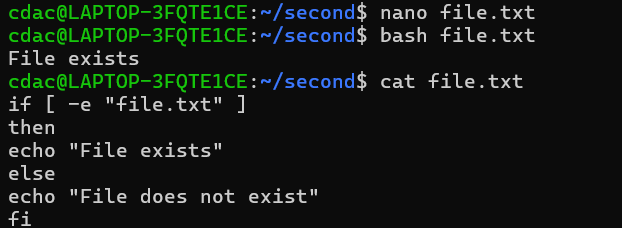
then

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 ]

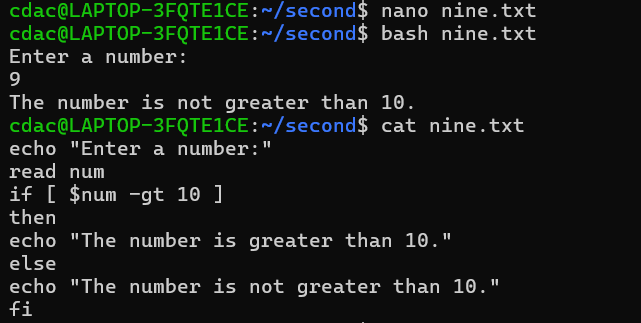
then

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 a in {1..10}

do

for b in {1..5}

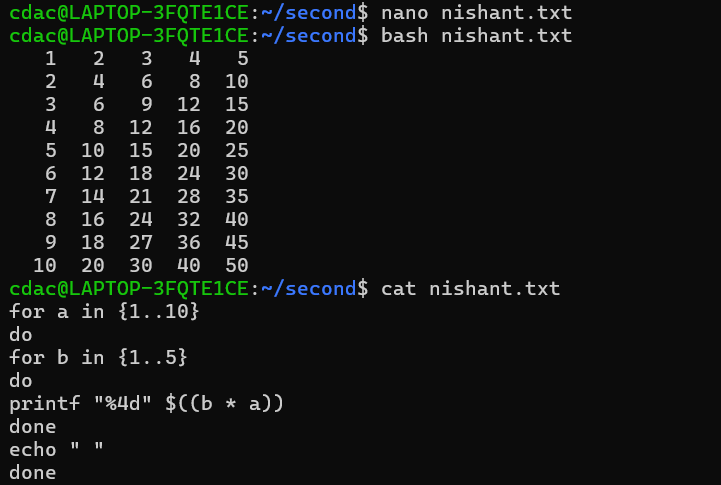
do

printf "%4d" $((b \* a))

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:"

read number

if [ $number -lt 0 ]; then

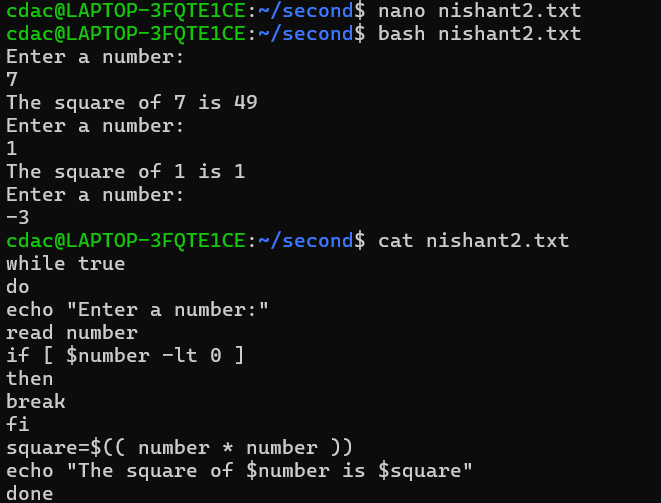
break

fi

square=$((number \* number))

echo "The square of $number is $square"

done



# **Part D**

**Common Interview Questions (Must know)**

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

:- Operating system is the computer system which helps user to interact with hardware of computer. It acts as a interface between user and hardware. Process management, memory management, file system management, device management, security, and user interface management are some of the functions of OS.

1. ***Explain the difference between process and thread.***

:- The process is the program loaded in the CPU for execution. Whereas thread is the small part of process. A process can me broken into number of sub tasks which are threads which run separately and collectively execute the whole function of process.

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

:- Virtual memory acts as a main memory but it’s in secondary memory and when a process is going in wait queue and if the queue is already filled then it will swap out into suspended wait block which is in virtual memory.

1. ***Describe the difference between multiprogramming, multitasking, and multiprocessing.***

:- Multiprogramming :- It is type of operating system in which multiple programs run in one processor simultaneously.

Multitasking:- In this multiple tasks are executed in simultaneous manner and it is done by single processor as well. It is basically Round-robin multi programming.

Multi-processing:- In this type of OS in which multiple CPUs are used to execute multiple programs at same time.

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

**:-** A file system in Operating system is a system which is used to store, organize and manage files and directories on storage device. Components of file system are files, directories, access control and permissions, matadata etc.

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

:- Deadlock is a phenomenon of operating system in which a processes cannot be executed because each of them holding a resource and waiting for another resource which has been already holding by other process. Conditions because of which deadlock occurs are mutual exclusion, hold and wait, no preemption and circular wait. If any of these conditions fails the deadlock can be prevented.

1. ***Explain the difference between a kernel and a shell.***

:- Kernel controls all the functions of hardware while shell allows users to interact with kernel. Kernel is the inner layer of OS. Shell is Outer layer of OS.

1. ***What is CPU scheduling, and why is it important?***

:- CPU scheduling is the process of deciding which program to run on the CPU at one time. This causes smoothly execution of processes without interfering each other. Therefore, It is very important.

1. ***How does a system call work?***

:- A system call is a type of interrupt when CPU requires hardware for I/O operations.

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

:- Software Helps to communicate Operating System with the hardware . It acts as a translator which allows OS and hardware to understand each other

1. ***Explain the role of the page table in virtual memory management.***

:- Page table is generated by memory management unit which contains the page location in the frame.

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

:-When computer spends more time to swap out data between main and secondary memory(paging) then Trashing happens.

It can be avoided by

-Better page replacement algorithms

-Increasing ram capacity

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

:- A semaphore is a synchronization primitive used in computer science to control access to a shared resource in a concurrent system, such as a multithreaded or multiprocess application. It is essentially a variable or abstract data type that is used to manage multiple processes or threads' access to resources.

1. ***How does an operating system handle process synchronization?***

:-Process synchronization helps to access shared resources to multiple processes It can be handled by multiple mechanisms like Semaphore signaling mechanism that helps control access to shared resources. Using a variable which can be 0 or 1. Mutex :- An object which ensures that only one process can access a shared resource at a time

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

:- A signal which is sent to CPU if an event requires immediate attention of CPU. Example Mouse Click.

1. ***Explain the concept of a file descriptor.***

:\_A file descriptor is an integer that uniquely identifies an open file or input/output (I/O) resource in a computer system, typically in Unix-like operating systems (such as Linux). It is used by the operating system to keep track of files, sockets, and other I/O resources that a process is working with.

1. ***How does a system recover from a system crash?***

:-Restarting processes and Rollback mechanisms can be the options to recover from system crash.

1. ***Describe the difference between a monolithic kernel and a microkernel.***

:\_Monolithic Kernel is a single large kernel that manages all system operations.

Microkernel is a minimal kernel that handles only essential tasks, with other services running in user space.

1. ***What is the difference between internal and external fragmentation?***

:- Internal fragmentation-occurs when a process size is smaller than the block size in main memory which leads to some empty space which is unused.

External fragmentation-occurs when a process size is bigger than the block size in main memory process need contiguous memory .so even though space is available it and can not accommodate the process.

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

:- OS manage I/O operations by generating interrupt and caching-keeping most frequently used data

and buffering-temporarily storing data.

1. ***Explain the difference between preemptive and non-preemptive scheduling.***

:- In Preemptive -CPU can be taken away from a running process voluntarily or if some higher priority process comes.

Example-Round Robin

In Non preemptive CPU cannot be taken away by a running process until its completion.

Example-FCFS(first come first serve)

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

:- Round-robin (RR) scheduling assigns each process a fixed time slice (quantum).

If a process doesn’t finish in its time slice, it is moved to the back of the queue.

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

:- In Process Scheduling Algorithms Processes are assigned a **priority number** (higher numbers may run first).

Types:

Preemptive: A higher-priority process can interrupt a lower-priority one.

Non-preemptive**:** A running process is not interrupted

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

:- SJN also known as Shortest Job First (SJF). It is algorithm which is used when the processes which can be executed firstly and swiftly because of they don’t need more time to use CPU. In SJN the process with the smallest CPU burst is executed first.

1. ***Explain the concept of multilevel queue scheduling.***

:- In this, processes are divided into multiple queues namely foreground, background and system. Each queue has its own scheduling algorithm.

Example:

Foreground: Uses Round Robin.

Background: Uses First Come First Serve (FCFS).

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

:- A Process Control Block (PCB) is a data structure that stores information about a process.  
 It contains:

1. Process ID (PID)
2. Process State (Ready, Running, Blocked)
3. CPU Registers (For resuming execution)
4. Program Counter (Next instruction to execute)
5. Memory Allocation (Page tables, stack, heap)
6. I/O Status (Files opened by the process)
7. ***Describe the process state diagram and the transitions between different process states.***

:- States in a process life cycle:

1. New – Process is created.
2. Ready – Process is waiting for CPU.
3. Running – Process is executing.
4. Waiting – Process is waiting for I/O.
5. Terminated – Process has completed execution.

Transitions:

* Ready → Running: When CPU is assigned.
* Running → Ready: If interrupted.
* Running → Waiting: Waiting for I/O.
* Waiting→ Ready: I/O is complete.
* Running → Terminated: Process finishes.

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

:- Processes communicate using Inter-Process Communication (IPC) mechanisms like pipes, message queues, shared memory, and sockets.

1. ***What is process synchronization, and why is it important?***

:- Process synchronization ensures that processes coordinate their actions when accessing shared resources to avoid conflicts and race conditions.

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

:- A zombie process occurs when a child process finishes execution, but its parent hasn't read its exit status yet, leaving it in the process table.

1. ***Describe the difference between internal fragmentation and external fragmentation.***

:- Internal Fragmentation: Unused memory within a memory allocation block.

External Fragmentation: Unused memory outside allocated blocks, causing inefficient memory usage.

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

:- Demand Paging loads pages into memory only when needed, which saves memory and reduces the time spent loading unused pages.

1. ***Explain the role of the page table in virtual memory management.***

:- The page table maps virtual memory addresses to physical memory addresses, allowing virtual memory to be used effectively.

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

:- The MMU translates virtual addresses to physical addresses using the page table during memory access.

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

:- Thrashing happens when excessive paging reduces system performance. It can be avoided by improving memory allocation, increasing physical memory, or optimizing page replacement algorithms.

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

:- A system call provides an interface for user programs to request services from the OS, such as file management or process control.

1. ***Describe the difference between a monolithic kernel and a microkernel.***

:- Monolithic Kernel is a single large kernel that manages all system operations.

Microkernel is a minimal kernel that handles only essential tasks, with other services running in user space.

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

:- The OS uses device drivers, buffers, and I/O schedulers to handle I/O requests efficiently.

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

:- A race condition occurs when multiple processes access shared resources concurrently in an unpredictable manner. It can be prevented using locks, semaphores, or other synchronization mechanisms.

1. ***Describe the role of device drivers in an operating system.***

***:-*** Device drivers manage communication between the OS and hardware devices, providing an abstraction layer so software can interact with hardware.

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

:- The zombie process is the process which completed its execution but is not terminated because it didn’t tell its status to its parent process. Zombie process can be prevented by system creates init process and becomes its parents helping it to terminate.

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

:- The orphan process is that process which’s parent is executed and terminated. So to handle such processes system generates init process which adopt these processes and becomes their parent.

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

:- A parent process creates a child process via fork(), and the parent controls the child’s execution and can wait for the child’s termination.

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

:- fork() creates a new process by duplicating the calling (parent) process, resulting in two identical processes: the parent and the child.

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

:- The parent process uses the wait() or waitpid() system calls to wait for the child process to finish and retrieve its exit status.

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

:- Exit status of child process give the information if the child process is executed successfully or not.

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

:- The parent can send signals to the child process or allow it to terminate on its own.

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

:- Process Group is a collection of processes that can be controlled together. While Session is a group of process groups associated with a controlling terminal.

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

:- The **exec()** family replaces the current process image with a new program, effectively running a new program in the context of the current process.

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

:- waitpid() waits for child process and gets the status of that child process. On the other side wait() only waits for child process without retrieving its status.

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

:- The process is terminated in OS after its execution. It happens after the termination signal generates. Then the resources of these processes are taken back from them by Kernel.

1. ***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?***

**:-** The LTS allows the processes to enter the ready state after its creation i.e from new state. It influences the degree of multiprogramming in an OS by managing number of processes in memory.

1. ***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?***

:- The frequency of STS of taking decisions to schedule which process first for execution is better than MTS and LTS.

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

:- When there is not any space for process to enter ready queue the Medium Term Scheduler swaps out the process from ready block to virtual memory and when ready queue gets freed it again swaps in that process from virtual memory to ready block.

# **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.***

**:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Response Time** | **Waiting Time** |
| **P1** | 0 | 5 | 0 | 0 |
| **P2** | 1 | 3 | 4 | 4 |
| **P3** | 2 | 6 | 6 | 6 |

Average Waiting Time =Total Waiting Time / Number of Processes

=10/3

= 3.33

1. ***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.***

**:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Response Time** | **Waiting Time** | **Turnaround Time** |
| **P1** | 0 | 3 | 0 | 0 | 3 |
| **P2** | 1 | 5 | 7 | 7 | 12 |
| **P3** | 2 | 1 | 1 | 1 | 2 |
| **P4** | 3 | 4 | 1 | 1 | 5 |

Average Turnaround Time =Total Turnaround Time / Number of Processes

=22/4

=5.5

1. ***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.***

***:-***

**Non-Preemptive:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Priority** | **Waiting Time** | **Turnaround Time** |
| **P1** | 0 | 6 | 3 | 0 | 6 |
| **P2** | 1 | 4 | 1 | 5 | 10 |
| **P3** | 2 | 7 | 4 | 10 | 19 |
| **P4** | 3 | 2 | 2 | 7 | 12 |

Average Waiting Time =Total Waiting Time / Number of Processes

=22/4

= 5.5

**Preemptive:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Priority** | **Waiting Time** | **Turnaround Time** |
| **P1** | 0 | 6 | 3 | 6 | 12 |
| **P2** | 1 | 4 | 1 | 0 | 4 |
| **P3** | 2 | 7 | 4 | 10 | 17 |
| **P4** | 3 | 2 | 2 | 2 | 4 |

Average Waiting Time =Total Waiting Time / Number of Processes

=18/4

= 4.5

1. ***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.***

**:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Waiting Time** | **Turnaround Time** |
| **P1** | 0 | 4 | 6 | 10 |
| **P2** | 1 | 5 | 8 | 13 |
| **P3** | 2 | 2 | 2 | 4 |
| **P4** | 3 | 3 | 7 | 10 |

**:-**

Average Turnaround Time =Total Turnaround Time / Number of Processes

=37/4

=9.25

1. ***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?***

**:-**

1. fork() is used to create copy of parent process and this process can be called as child process of that parent process.
2. As per in question parent process is with x=5. After forking the parent process will be incremented by 1 and x will become 6.
3. In the case with child process which is exact replica of parent process initially also has x with 5 and after incrementing by 1 , here also x becomes 6.