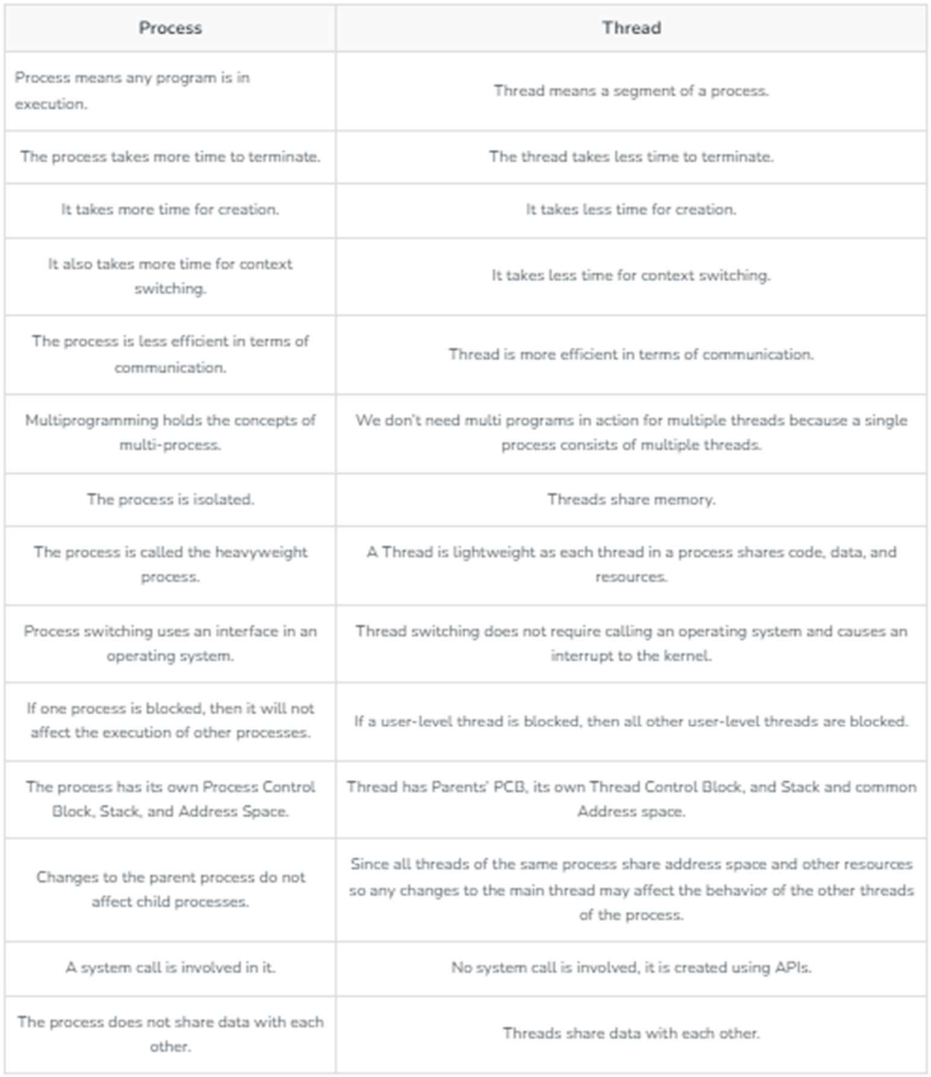
1. What is an operating system, and what are its primary functions?

   -→ An operating system (OS) is an interface between the computer hardware and the user, managing software resources and computer hardware. The primary functions of an operating system are process management, memory management, file systems management, device management, and security and privacy.

2. Explain the difference between process and thread.



Process	Thread
Process means any program is in execution.	Thread means a segment of a process.
The process takes more time to terminate.	The thread takes less time to terminate.
It takes more time for creation.	It takes less time for creation.
It also takes more time for context switching.	It takes less time for context switching.
The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
The process is isolated.	Threads share memory.
The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
If one process is blocked, then it will not affect the execution of other processes.	If a user-level thread is blocked, then all other user-level threads are blocked.
The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.
A system call is involved in it.	No system call is involved, it is created using APIs.
The process does not share data with each other.	Threads share data with each other.

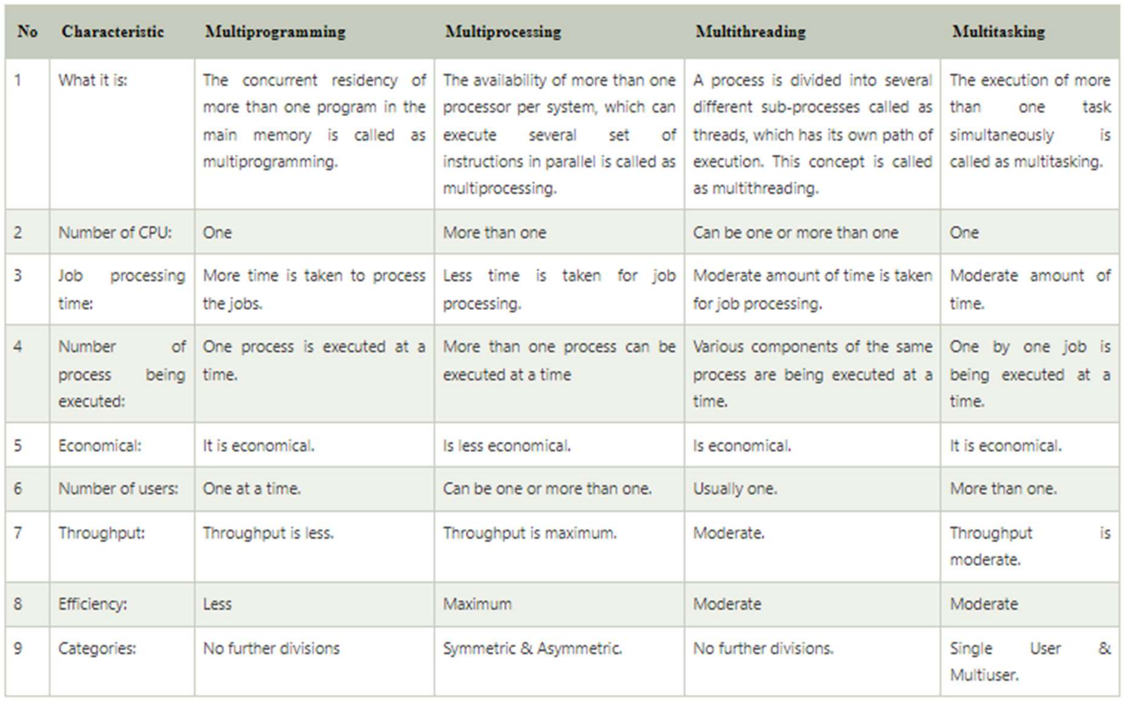
→

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

→ Its a memory space in hard-drive, which work like physical memory to entertain large processes whose size is bigger than the RAM. It is an illusion memory. In Virtual Memory process is divided into fixed sized partions know as pages. OS loads the process's pages from virtual memory to physical memory on demand of CPU i.e. known as Demand Paging. While working with virtual memory pages are saved as per their logical address. While loading the pages from virtual memory to physical memory the pages get physical address. If a page loaded from virtual memory to physical memory i.e. know as swap-in process.

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

→



No	Characteristic	Multiprogramming	Multiprocessing	Multithreading	Multitasking
1	What it is:	The concurrent residency of more than one program in the main memory is called as multiprogramming.	The availability of more than one processor per system, which can execute several set of instructions in parallel is called as multiprocessing.	A process is divided into several different sub-processes called as threads, which has its own path of execution. This concept is called as multithreading.	The execution of more than one task simultaneously is called as multitasking.
2	Number of CPU:	One	More than one	Can be one or more than one	One
3	Job processing time:	More time is taken to process the jobs.	Less time is taken for job processing.	Moderate amount of time is taken for job processing.	Moderate amount of time.
4	Number of process being executed:	One process is executed at a time.	More than one process can be executed at a time	Various components of the same process are being executed at a time.	One by one job is being executed at a time.
5	Economical:	It is economical.	Is less economical.	Is economical.	It is economical.
6	Number of users:	One at a time.	Can be one or more than one.	Usually one.	More than one.
7	Throughput:	Throughput is less.	Throughput is maximum.	Moderate.	Throughput is moderate.
8	Efficiency:	Less	Maximum	Moderate	Moderate
9	Categories:	No further divisions	Symmetric & Asymmetric.	No further divisions.	Single User & Multiuser.

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

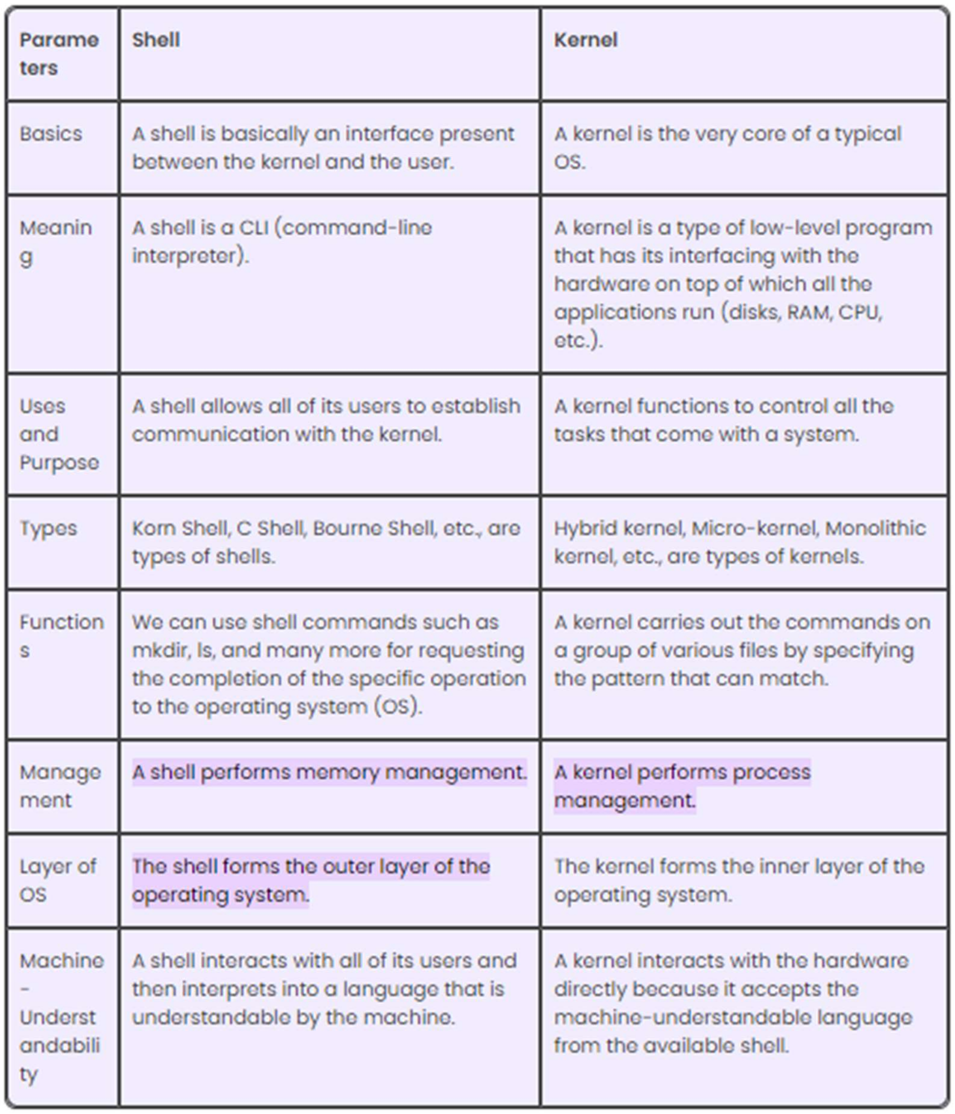
→ A file system is a structure used by an operating system to organize and manage files on a storage device such as a hard drive, solid state drive (SSD), or USB flash drive. It defines how data is stored, accessed, and organized on the storage device.

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

→A deadlock occurs when a process or multiple processes are unable to continue running because they are waiting for resources that are currently held by other processes.

Deadlock can be prevented by eliminating any of the four necessary conditions, which are mutual exclusion, hold and wait, no preemption, and circular wait

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



Parameters	Shell	Kernel
Basics	A shell is basically an interface present between the kernel and the user.	A kernel is the very core of a typical OS.
Meaning	A shell is a CLI (command-line interpreter).	A kernel is a type of low-level program that has its interfacing with the hardware on top of which all the applications run (disks, RAM, CPU, etc.).
Uses and Purpose	A shell allows all of its users to establish communication with the kernel.	A kernel functions to control all the tasks that come with a system.
Types	Korn Shell, C Shell, Bourne Shell, etc., are types of shells.	Hybrid kernel, Micro-kernel, Monolithic kernel, etc., are types of kernels.
Functions	We can use shell commands such as mkdir, ls, and many more for requesting the completion of the specific operation to the operating system (OS).	A kernel carries out the commands on a group of various files by specifying the pattern that can match.
Management	A shell performs memory management.	A kernel performs process management.
Layer of OS	The shell forms the outer layer of the operating system.	The kernel forms the inner layer of the operating system.
Machine - Understandability	A shell interacts with all of its users and then interprets into a language that is understandable by the machine.	A kernel interacts with the hardware directly because it accepts the machine-understandable language from the available shell.

→

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

→ CPU scheduling is the process of deciding which processes should be executed on a CPU, and in what order. It's important because it directly impacts a system's performance, resource utilization.

- Maximizes CPU utilization: CPU scheduling ensures that the CPU is being used as much as possible, so the computer is more productive
- Minimizes waiting time: CPU scheduling aims to minimize the time that processes spend waiting to be executed.
- Fairness: CPU scheduling aims to be fair to all processes.

9. How does a system call work?

→ A system call is a way for a program to request services from an operating system's kernel.

System calls allow programs to access privileged functionalities provided by the operating system. They also ensure that hardware resources are isolated from user space processes, and prevent direct access to the kernel or hardware memory

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

→ The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write higher-level application code independently of whatever specific hardware the end-user is using.

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

→ A Page Table is a data structure used by the operating system to keep track of the mapping between virtual addresses used by a process and the corresponding physical addresses in the system's memory. A Page Table Entry (PTE) is an entry in the Page Table that stores information about a particular page of memory.

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

→ **Thrashing** is a condition or a situation when the system is spending a major portion of its time servicing the page faults, but the actual processing done is very negligible.

What are the causes of thrashing?
The high degree of multiprogramming, lack of frames, and page replacement policy are the main causes of thrashing.

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

→ A semaphore is a variable or abstract data type used to control access to a shared resource by multiple threads in a computer system. Semaphores are a type of synchronization primitive that help prevent critical section problems and race conditions

14. How does an operating system handle process synchronization?

→ Allowing the processes to work on or access (read / write) to a resource (memory) one by one is called process synchronization

Process synchronization is the only solution to resolve the Producer-Consumer Problem

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

→

What is an interrupt? An interrupt is a signal emitted by a device attached to a computer or from a program within the computer. It requires the operating system (OS) to stop and figure out what to do next. An interrupt temporarily stops or terminates a service or a current process.

16. Explain the concept of a file descriptor.

→ A file descriptor is a unique identifier that an operating system assigns to a file when it's opened. It's used by programs to interact with files, sockets, and other input/output (I/O) resources

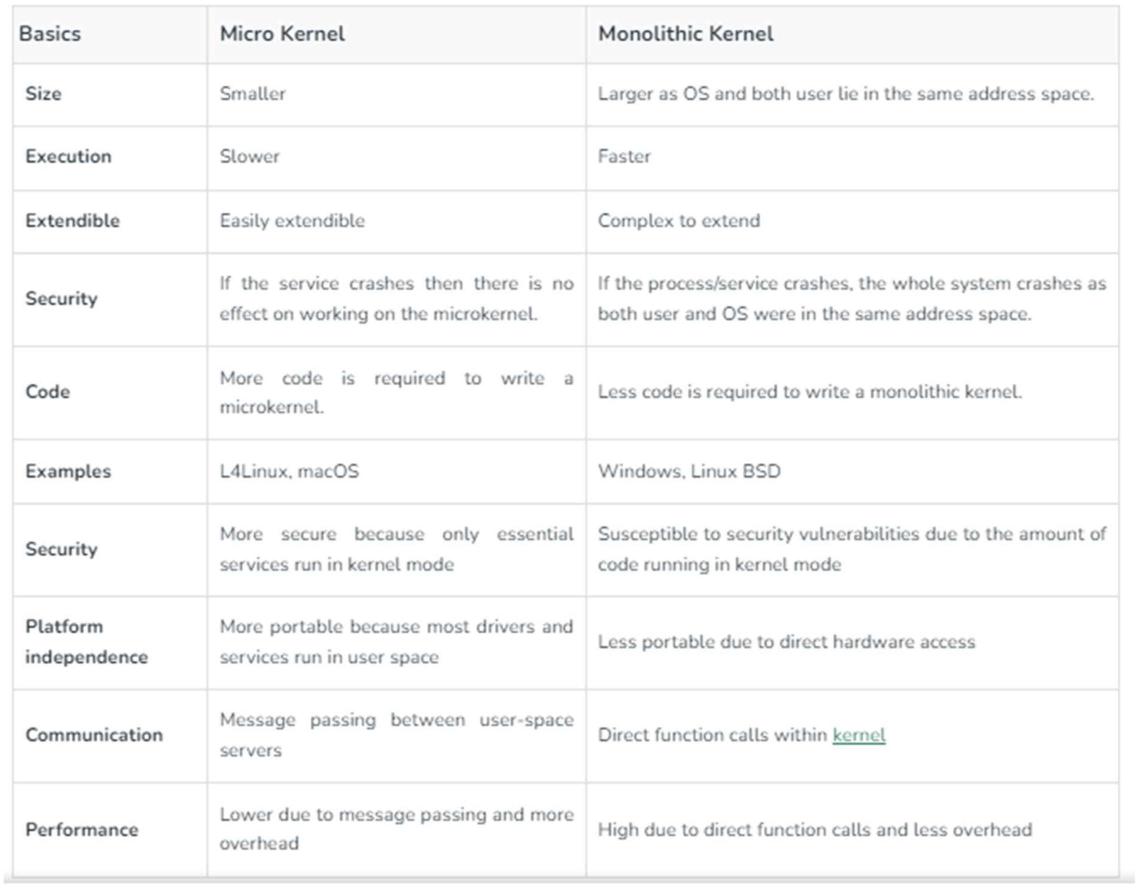
A file descriptor is a unique identifier or reference that the operating system assigns to a file when it is opened. It allows programs to interact with files, sockets, or other input/output (I/O) resources. The file descriptor is used by the operating system to keep track of the file and perform operations on it.

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

→ The first step to recover a system after a crash is to identify the cause of the problem. This can help you narrow down the possible solutions and prevent future crashes. Some common causes of system crashes are hardware failures, software bugs, malware infections, power outages, and human errors.

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

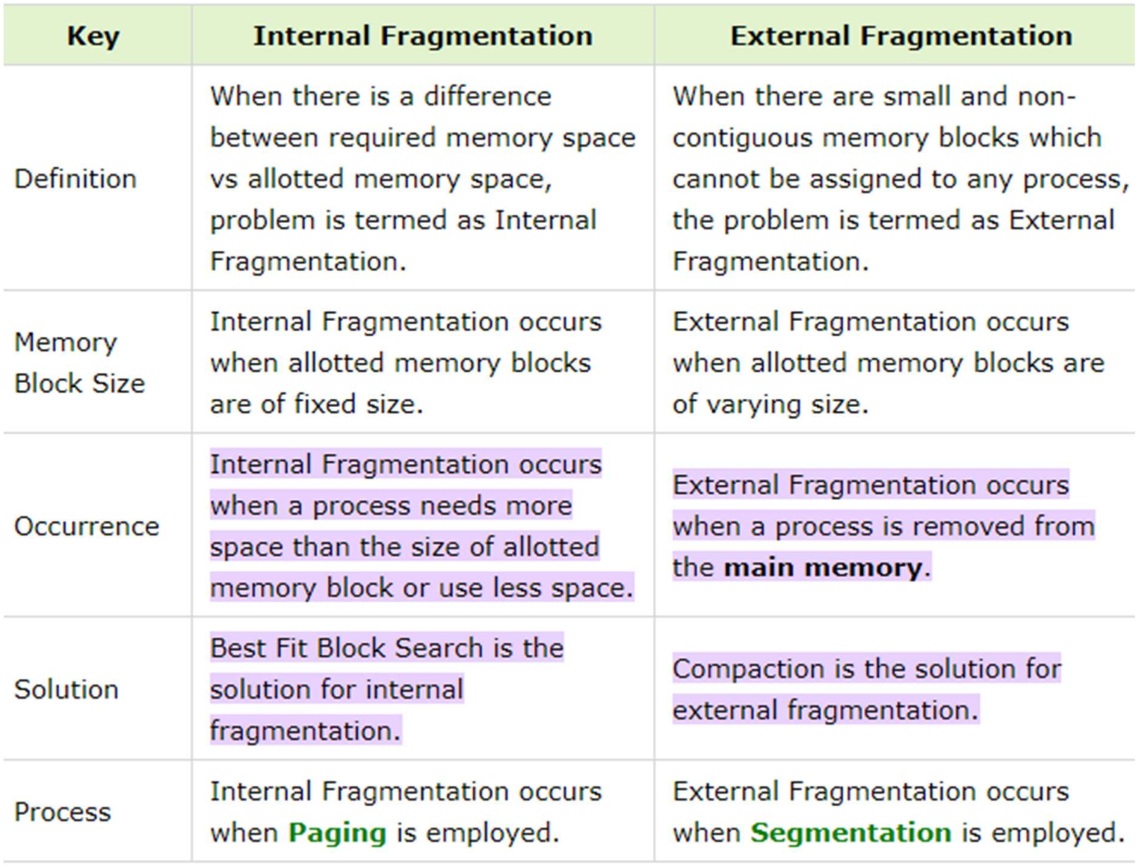
→



Basics	Micro Kernel	Monolithic Kernel
Size	Smaller	Larger as OS and both user lie in the same address space.
Execution	Slower	Faster
Extendible	Easily extendible	Complex to extend
Security	If the service crashes then there is no effect on working on the microkernel.	If the process/service crashes, the whole system crashes as both user and OS were in the same address space.
Code	More code is required to write a microkernel.	Less code is required to write a monolithic kernel.
Examples	L4Linux, macOS	Windows, Linux BSD
Security	More secure because only essential services run in kernel mode	Susceptible to security vulnerabilities due to the amount of code running in kernel mode
Platform independence	More portable because most drivers and services run in user space	Less portable due to direct hardware access
Communication	Message passing between user-space servers	Direct function calls within kernel
Performance	Lower due to message passing and more overhead	High due to direct function calls and less overhead

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

→

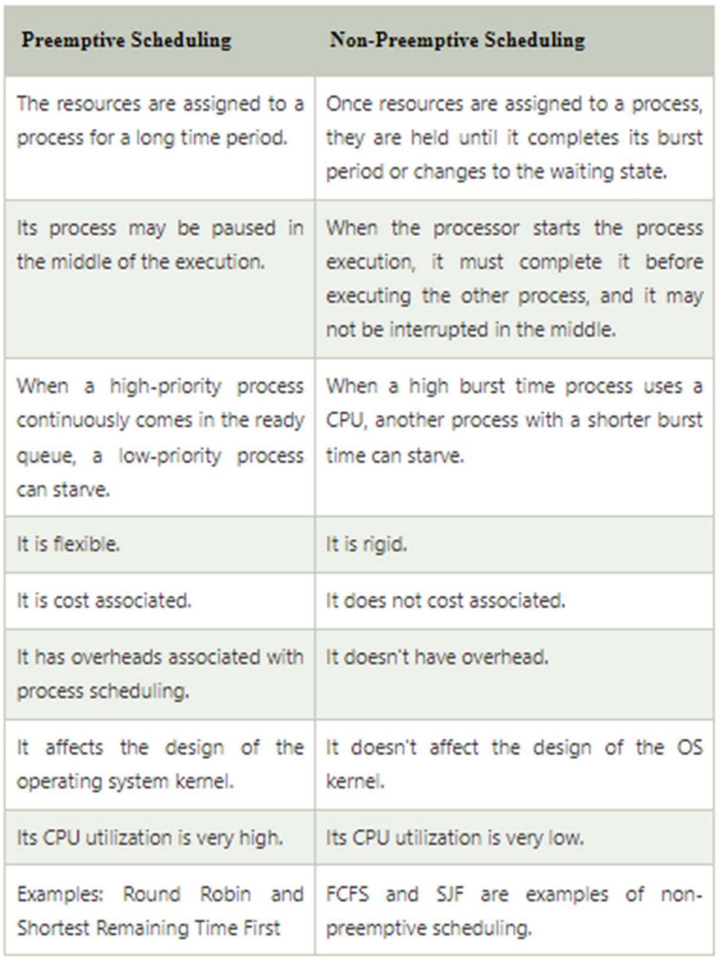


Key	Internal Fragmentation	External Fragmentation
Definition	When there is a difference between required memory space vs allotted memory space, problem is termed as Internal Fragmentation.	When there are small and non-contiguous memory blocks which cannot be assigned to any process, the problem is termed as External Fragmentation.
Memory Block Size	Internal Fragmentation occurs when allotted memory blocks are of fixed size.	External Fragmentation occurs when allotted memory blocks are of varying size.
Occurrence	Internal Fragmentation occurs when a process needs more space than the size of allotted memory block or use less space.	External Fragmentation occurs when a process is removed from the **main memory**.
Solution	Best Fit Block Search is the solution for internal fragmentation.	Compaction is the solution for external fragmentation.
Process	Internal Fragmentation occurs when **Paging** is employed.	External Fragmentation occurs when **Segmentation** is employed.

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

→ An operating system (OS) manages input/output (I/O) operations by: Providing interfaces ,Handling data transfer, Using device drivers, Using interrupts, Using buffering, Using direct memory access (DMA), Using caching and spooling:

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

→ 

Preemptive Scheduling	Non-Preemptive Scheduling
The resources are assigned to a process for a long time period.	Once resources are assigned to a process, they are held until it completes its burst period or changes to the waiting state.
Its process may be paused in the middle of the execution.	When the processor starts the process execution, it must complete it before executing the other process, and it may not be interrupted in the middle.
When a high-priority process continuously comes in the ready queue, a low-priority process can starve.	When a high burst time process uses a CPU, another process with a shorter burst time can starve.
It is flexible.	It is rigid.
It is cost associated.	It does not cost associated.
It has overheads associated with process scheduling.	It doesn't have overhead.
It affects the design of the operating system kernel.	It doesn't affect the design of the OS kernel.
Its CPU utilization is very high.	Its CPU utilization is very low.
Examples: Round Robin and Shortest Remaining Time First	FCFS and SJF are examples of non-preemptive scheduling.

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

→ Round-robin scheduling is a method for allocating time or resources in a circular, systematic, and balanced way.allocates each task an equal share of the CPU time.

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

→ The priority scheduling algorithm is a CPU scheduling algorithm that assigns priorities to processes and executes them based on those priorities. The algorithm's goal is to optimize system performance and ensure critical tasks are executed on time.

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

→ The shortest job next (SJN) scheduling algorithm, also known as shortest job first (SJF), is a scheduling policy that chooses the waiting process with the shortest execution time to execute next. It's used to optimize CPU utilization and reduce average waiting time.

25. Explain the concept of multilevel queue scheduling.

→ Multilevel queue scheduling is a CPU scheduling algorithm used in operating systems that organizes processes into different queues based on their characteristics or priority. Each queue has its own scheduling algorithm, and there's a scheduling system among the queues themselves.

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

→ A Process Control Block (PCB) is a data structure used by the operating system to store all the information about a process, including its state, program counter, CPU registers, memory allocation, and scheduling information.

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

→. the main memory is full and a process with higher priority needs to be executed, the process transitions from the ready state to the suspend ready state. This transition frees up space in the ready state for the higher-priority process by moving a lower-priority process to the suspend ready state.

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

→ The communication between these processes can be performed using shared memory or message passing. In shared memory, a common region of memory is shared by multiple processes. Each process can read from or write to this shared memory area.

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

→ Process synchronization is the process of coordinating multiple processes in an operating system to share resources and data. It's important because it prevents errors, inconsistencies, and deadlocks that can occur when processes interfere with each other.

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

→ A zombie process is a process that has finished running but still has an entry in the process table. This happens when a child process finishes running, but its parent process doesn't collect its termination status.

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

→

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

→Demand paging is a memory management technique that loads data from disk into RAM when it's accessed. It's used by operating systems to efficiently use system resources. Demand paging improves memory management efficiency by: Reducing startup time, Loading only necessary pages, Reducing memory requirements, Freeing up space

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

→ A Page Table is a data structure used by the operating system to keep track of the mapping between virtual addresses used by a process and the corresponding physical addresses in the system's memory; A page table is a data structure that maps virtual addresses to physical addresses in a computer's virtual memory system

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

→ A memory management unit (MMU) translates virtual memory addresses into physical addresses in main memory. This process is called address translation. The MMU uses translation tables stored in main memory to perform this task.

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

→ Thrashing is a computing term that describes a situation when a computer system spends too much time on non-productive tasks, which can result in poor performance. It can occur when a system is overwhelmed by demand, lacks resources, or relies heavily on virtual memory.

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

→ A system call is a way for a computer program to request services from the operating system (OS) it's running on

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

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

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

→A race condition occurs when multiple processes or threads access shared data at the same time, resulting in unexpected outcomes.

Race condition vulnerabilities are a serious threat to computer systems and applications. They can cause significant damage, such as data corruption, resource starvation, or denial of service attacks.

To prevent race conditions, it is crucial to use locking mechanisms such as semaphores, mutexes & atomic operations

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

→ Device drivers are necessary to enable a computer to interface and interact with specific devices. A device driver's role in an operating system is to act as a bridge between a computer's hardware and its operating system and applications

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

→ A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table.

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

→ An orphan process is a computer process that's still running, but its parent process has terminated or finished. The operating system (OS) handles orphan processes by assigning them to the init process, which then becomes the orphan process's new parent.

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

→ . A child process results from the parent calling the fork() primitive and duplicating its code to create a child. The PID of the child is returned to the parent process so that it can talk to it. Each child has its parent's identifier, the PPID.

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

→ The fork() system call is used in Unix-based OS to create a new process by duplicating the calling process. The new process is an exact copy of the parent process, with its own address space and memory.2

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

→ This can be done by using wait() system call. The parent process may then issue a wait() system call, which suspends the execution of the parent process while the child executes and when the child finishes execution, it returns exit status to operating system.

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

→ A call to wait() blocks the calling process until one of its child processes exits or a signal is received.

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

→ I have a process which spawns exactly one child process. Now when the parent process exits for whatever reason (normally or abnormally, by kill, ^C, assert failure or anything else) I want the child process to die. How to do that correctly?

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

→A process group is a collection of one or more related processes. A session is a collection of one or more process groups, possibly with a controlling terminal.

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

→ The exec family of functions shall replace the current process image with a new process image. The new image shall be constructed from a regular, executable file called the new process image file.

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

→ Suspends the calling process until a child process ends or is stopped. More precisely, waitpid() suspends the calling process until the system gets status information on the child. If the system already has status information on an appropriate child when waitpid() is called, waitpid() returns immediately.

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

→ In Unix-like systems, all process termination requests are handled using signals. The kill function takes the Process ID and signal to send, and is accessible with the kill command.

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?

→ The long-term scheduler is a key component in the process scheduling hierarchy that controls the degree of multiprogramming in an operating system
by selecting processes to load 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?

→ Short term is also a minimal time sharing system. The long term selects the processes from the pool and loads them into memory for execution. Medium term can reintroduce the process into memory and execution can be continued. Short term selects those processes that are ready to execute.

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

→ Aging of process: The medium-term scheduler can adjust the priority of a process based on how long it has been waiting for execution. This is known as the aging of process, which ensures that processes that have been waiting for a long time are given priority over newer processes