Explain the difference between a multiprogramming and a time sharing system.

Multiprogramming System	Time-Sharing System
A multiprogramming operating system allows you to run numerous processes simultaneously by monitoring their states and switching between them.	Time sharing is the logical expansion of multiprogramming. In this time-share operating system, many users/processes are assigned with IT resources in different time slots.
The problem of underutilization of the processor and memory has been remedied, and the CPU can now run numerous programs. This is why it's known as multiprogramming.	The processor's time is divided among several users. It's dubbed a time-sharing operating system for this reason.
A single processor can run the process in multiprogramming.	Two or more users can use a CPU in their terminal in this operation.
There is no set time slice in a multiprogramming OS.	A fixed time slice exists in a time-sharing OS.
The executive power in a multiprogramming OS system is not delegated before a task is completed.	Executive power is switched off in a time-sharing OS system before execution is completed.
The system does not waste time working on many processes here.	Each process takes the same amount of time or less time here.
The system relies on devices to switch between activities in Multiprogramming OS, such as I/O interrupts.	The operating system uses the time to transition between processes in time-sharing.
Multiple programs are the system model of a multiprogramming system.	Multiple programs and users are part of the time-sharing system's system model.
The multiprogramming system maximizes Response time.	The system of time-sharing maximizes Response time.
Mac OS, Windows OS, and microcomputers like MP/M, XENIX, and ESQview are just a few examples.	Windows NT server, Unix, Linux, Multics, TOPS-10, and TOPS-20 are some examples.

Suggest a protocol which can violate circular wait condition of deadlock and prove its worthiness.

We define a one-to-one function $F: R \rightarrow N$, where N is the set of natural numbers.

We can require that a process requesting an instance of resource type R_j must have released any resources R_i such that $F(R_i) >= F(R_j)$. If several instances of the same resource type are needed, a single request for all of them must be issued.

Proof by contradiction:

Let the set of processes involved in the circular wait be $\{P_0, P_1, \ldots, P_n\}$, where P_i is waiting for a resource R_i, which is held by process P_{i+1}. Then, since process P_{i+1} is holding resource R_i while requesting resource R_{i+1}, we must have $F(R_i) < F(R_{i+1})$ for all i. But this condition means that $F(R_0) < F(R_1) < \ldots < F(R_n) < F(R_0)$. By transitivity, $F(R_0) < F(R_0)$, which is impossible. Therefore, there can be no circular wait.

What is a thread and why is it called a light weight process

A thread is a lightweight process that can run concurrently with other threads in the same process. Threads share the same address space and resources as the process they belong to, but they have their own stack and registers. This allows threads to execute independently of each other, without interfering with each other.

Threads are called lightweight processes because they are much less expensive to create and manage than processes. This is because threads do not need to create their own address space or resources. They simply share the address space and resources of the process they belong to.

How does a dynamic memory binding technique differ from static memory binding technique? Explain the difference between a paging and segmentation in brief.

Static Binding	**Dynamic Binding**
It happens at the compile time.	It happens at the run time.
It is also called early binding.	It is also called late binding.
When all the information needed to call a function is available at the compile time, static binding occurs.	When the compiler cannot determine all the information to resolve the function call, dynamic binding occurs.
It can be achieved during normal function calls, function overloading and operator overloading.	It is achieved with the use of virtual functions.
Execution becomes faster than dynamic binding because the function call gets resolved before run time.	As the function call is resolved at run time, sometimes it leads to slower code execution.
It provides less flexibility compared to the dynamic binding.	It provides more flexibility as different types of objects at runtime can be handled by a single function call making the source code more readable.

Paging	Segmentation
In paging, the program is divided into fixed size pages.	In segmentation, the program is divided into variable size sections.
For the paging operating system is accountable.	For segmentation compiler is accountable.
Page size is determined by hardware.	Here, the section size is given by the user.
It is faster in comparison to segmentation.	Segmentation is slow.
Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.
The page table is employed to keep up the page data.	Section Table maintains the section data.
In paging, the operating system must maintain a free frame list.	In segmentation, the operating system maintains a list of holes in the main memory.
Paging is invisible to the user.	Segmentation is visible to the user.
A page is referred to as a physical unit of information.	A segment is referred to as a logical unit of information.
Paging results in a less efficient system.	Segmentation results in a more efficient system.

Explain different operations on a process.

1. Creation: This is the initial step of process execution activity. Process creation means the construction of a new process for the execution. This might be performed by system, user or old process itself.

2. Scheduling/Dispatching: The event or activity in which the state of the process is changed from ready to running. It means the operating system puts the process from ready state into the running state. Dispatching is done by operating system when the resources are free or the process has higher priority than the ongoing process.

3. Blocking: Block mode is basically a mode where process waits for input-output. Hence on the demand of process itself, operating system blocks the process and dispatches another process to the processor. Hence, in process blocking operation, the operating system puts the process in 'waiting' state.

4. Preemption: When a timeout occurs that means the process hadn't been terminated in the allotted time interval and next process is ready to execute, then the operating system preempts the process. This operation is only valid where CPU scheduling supports preemption. Basically this happens in priority scheduling where on the incoming of high priority process the ongoing process is preempted. Hence, in process preemption operation, the operating system puts the process in ‘ready’ state.

5. Termination: Process termination is the activity of ending the process. In other words, process termination is the relaxation of computer resources taken by the process for the execution.

Explain the role of process control block.

- **Interrupt handling:** The PCB also contains information about the interrupts that a process may have generated and how they were handled by the operating system.
- **Context switching:** The process of switching from one process to another is called context switching. The PCB plays a crucial role in context switching by saving the state of the current process and restoring the state of the next process.
- **Real-time systems:** Real-time operating systems may require additional information in the PCB, such as deadlines and priorities, to ensure that time-critical processes are executed in a timely manner.
- **Virtual memory management:** The PCB may contain information about a process’s virtual memory management, such as page tables and page fault handling.
- **Inter-process communication:** The PCB can be used to facilitate inter-process communication by storing information about shared resources and communication channels between processes.
- **Fault tolerance:** Some operating systems may use multiple copies of the PCB to provide fault tolerance in case of hardware failures or software errors.

What is multiprocessing system? Write its advantages and disadvantages over personal system.

A multiprocessing system is a computer system with two or more central processing units (CPUs). Each CPU can run its own processes simultaneously, which can improve the overall performance of the system.

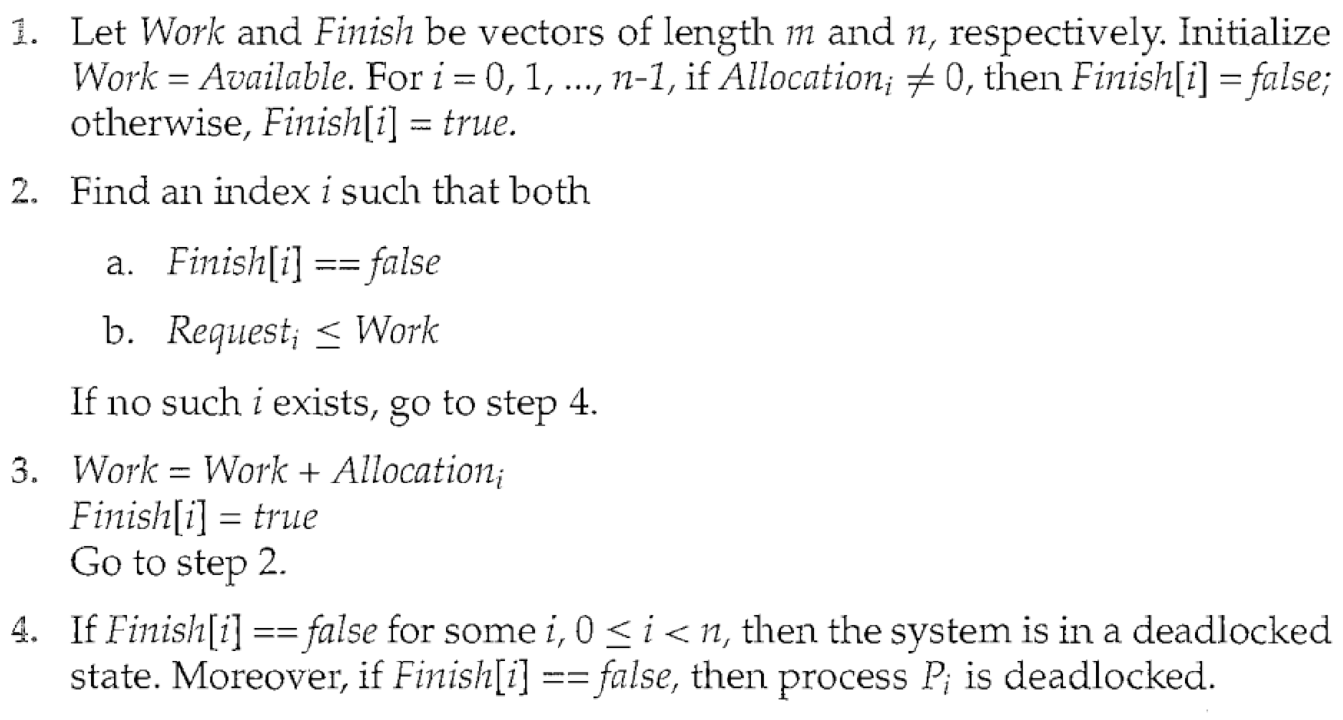
Here are some of the advantages of multiprocessing systems:

- **Increased performance:** Multiprocessing systems can improve the performance of applications.
- **Improved scalability:** Multiprocessing systems can be scaled up by adding more CPUs.
- **Improved availability:** Multiprocessing systems can continue to operate even if one of the CPUs fails.

Here are some of the disadvantages of multiprocessing systems:

- **Increased complexity:** Multiprocessing systems are more complex to design and implement than single-processor systems. This can lead to problems such as deadlocks and race conditions.
- **Increased cost:** Multiprocessing systems are more expensive than single-processor systems. This is because they require more hardware, such as CPUs and memory.
- **Incompatibility:** Multiprocessing systems may not be compatible with all applications.

Write an algorithm for deadlock detection when multiple instances of resources are there.

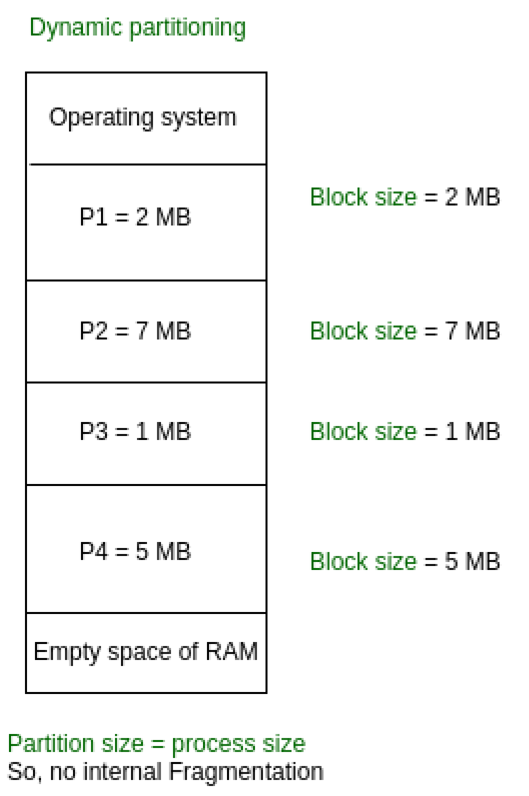


1. Let *Work* and *Finish* be vectors of length m and n, respectively. Initialize $Work = Available$. For $i = 0, 1, \ldots, n-1$, if $Allocation_i \neq 0$, then $Finish[i] = false$; otherwise, $Finish[i] = true$.
2. Find an index i such that both
   a. $Finish[i] == false$
   b. $Request_i \leq Work$

   If no such i exists, go to step 4.
3. $Work = Work + Allocation_i$
   $Finish[i] = true$
   Go to step 2.
4. If $Finish[i] == false$ for some i, $0 \leq i < n$, then the system is in a deadlocked state. Moreover, if $Finish[i] == false$, then process P_i is deadlocked.

Explain variable partitioning approach for contiguous allocation of memory.

1. Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
2. The size of partition will be equal to incoming process.
3. The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
4. Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory's size.



What do you mean by internal and external memory fragmentation?

Internal fragmentation	External fragmentation
In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to the method.
Internal fragmentation happens when the method or process is smaller than the memory.	External fragmentation happens when the method or process is removed.
The solution of internal fragmentation is the best-fit block.	The solution to external fragmentation is compaction and paging.
Internal fragmentation occurs when memory is divided into fixed-sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, which is called External fragmentation.
Internal fragmentation occurs with paging and fixed partitioning.	External fragmentation occurs with segmentation and dynamic partitioning.

It occurs on the allocation of a process to a partition greater than the process's requirement. The leftover space causes degradation system performance.	It occurs on the allocation of a process to a partition greater which is exactly the same memory space as it is required.
It occurs in worst fit memory allocation method.	It occurs in best fit and first fit memory allocation method.

Explain the difference between tree based and graph based directory structures.

Feature	**Tree Based Structure**	**Graph Based Structure**
Organization	Hierarchical	Not hierarchical
Sharing	Files cannot be shared between users	Files can be shared between users
Ease of searching	Easy to search	More difficult to search
Ease of deletion	Easy to delete files	More difficult to delete files
Dangling links	No dangling links	Can have dangling links
Scalability	Scalable	Less scalable
Flexibility	Less flexible	More flexible