OPERATING SYSTEMS

Unit1_Unit2_Unit3:Revision Class #7 Typical Concepts and QnAs related to Operating Systems

Nitin V Pujari
Faculty, Computer Science
Dean , IQAC, PES University

OPERATING SYSTEMS

Course Syllabus => Unit 1

Class No.	Chapter Title / Reference Literature	Topics to be covered	% of Portions covered	
			Reference chapter	Cumulative
1		Introduction: What Operating Systems Do, Computer-System Organization	1.1, 1.2	21.4
2		Computer-System Architecture, Operating-System Structure & Operations	1.3,1.4,1.5	
3		Kernel Data Structures, Computing Environments	1.10, 1.11	
4		Operating-System Services, Operating-System Design and Implementation	2.1, 2.6	
5		Process concept: Process in memory, Process State, Process Control Block, Context switch, Process Creation & Termination,	3.1 - 3.3	
6		CPU Scheduling - Preemptive and Non-Preemptive, Scheduling Criteria, FIFO Algrorithm	5.1-5.2	
7		Scheduling Algorithms:SJF, Round-Robin and Priority Scheduling	5.3	
8		Multi-Level Queue, Multi-Level Feedback Queue	5.3	
9		Multiprocessor and Real Time Scheduling	5.5, 5.6	
10		Case Study: Linux/ Windows Scheduling Policies.	5.7	
11		Inter Process Communication - Shared Memory, Messages	3.4	
12.		Named and unnamed pipes (+Review)	3.6.3	

OPERATING SYSTEMS

Course Syllabus => Unit 2

13	Introduction to Threads, types of threads, Multicore Programming, Multithreading Models	4.1 - 4.3	42.8
14	Thread creation, Thread Scheduling	5.4	
15	Pthreads and Windows Threads	4.4	
16	Mutual Exclusion and Synchronization: software approaches,	6.1-6.2	
17	principles of concurrency, hardware support	6.3-6.4	
18	Mutex Locks, Semaphores	6.5, 6.6	
19	Classic problems of Synchronization: Bounded-Buffer Problem, Readers-Writers problem	6.7-6.8	
20	Dining-Philosophers Problem	6.8	
21	Synchronization Examples: Synchronisation mechanisms provided by Linux/Windows/Pthreads.	6.9	
22	Deadlocks: principles of deadlock, Deadlock Characterization	7.1-7.3	
23	Deadlock Prevention, Deadlock example	7.4-7.5	
24	Deadlock Detection, Algorithm	7.6	

OPERATING SYSTEMS

Course Syllabus => Unit 3

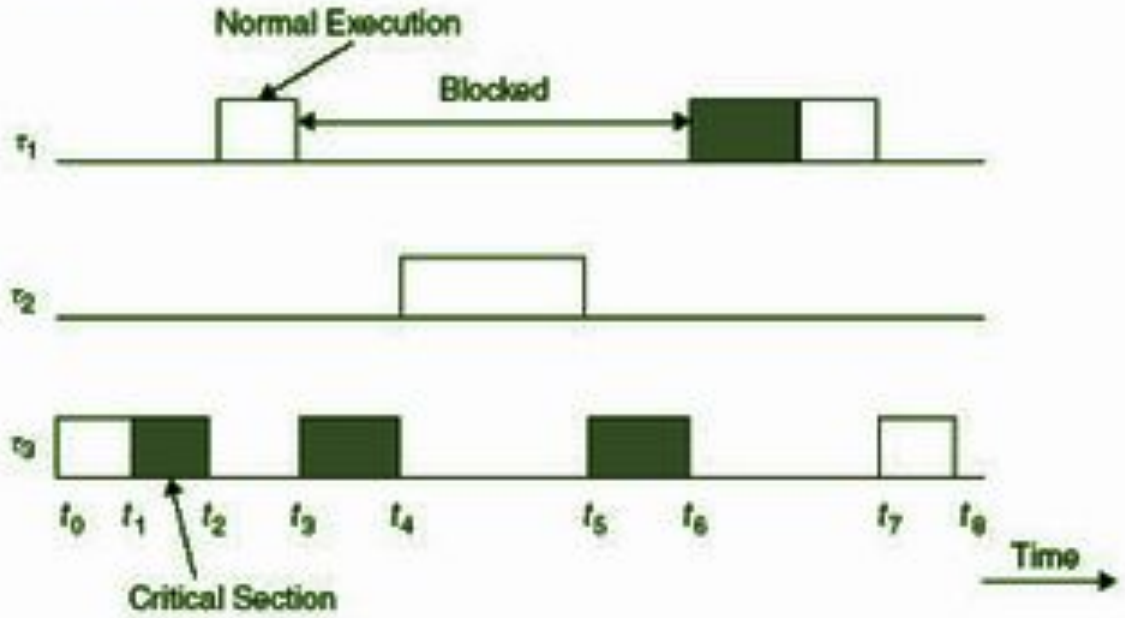
25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives - Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy - LRU etc. (in comparison with FIFO and Optimal)	9.4	
34	Page Replacement (contd.), Frame allocation	9.4,9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

Priority Inversion versus Priority Inheritance

- Both of these phenomena happen in priority scheduling in general and Preemptive Priority Scheduling in particular.
- **Priority Inversion** is a **problem** while **Priority Inheritance** using **Priority Ceiling** is a **solution**.
- Priority Inversion means that priority of tasks get **inverted**
- Priority Inheritance means that priority of tasks get **inherited**.
- Basically, in Priority Inversion, higher priority task (H) ends up waiting for middle priority task (M), when H is sharing critical section with lower priority task (L) and L is already in critical section.
- Effectively, H waiting for M results in **inverted priority** i.e. **Priority Inversion**. One of the solution for this problem is Priority Inheritance.
- In Priority Inheritance, **when L is in critical section**, **L inherits priority of H** at the time when H starts waiting for critical section. By doing so, M doesn't interrupt L and H doesn't wait for M to finish.
- Please note that inheriting of priority is done **temporarily** i.e. L goes back to its old priority when L comes out of critical section.

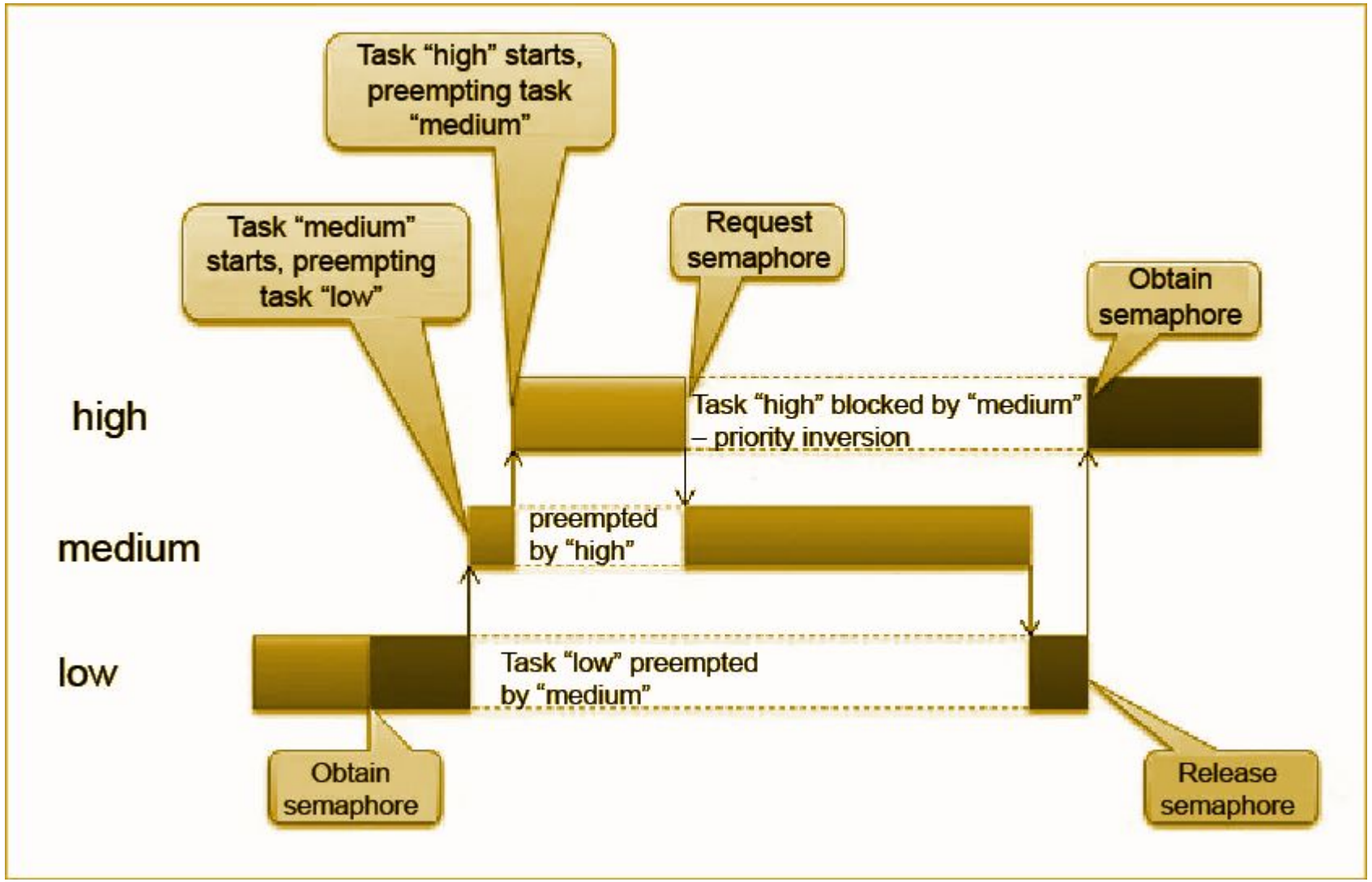
Priority Inversion versus Priority Inheritance

When a low-priority task blocks a higher-priority one, a priority inversion is said to occur

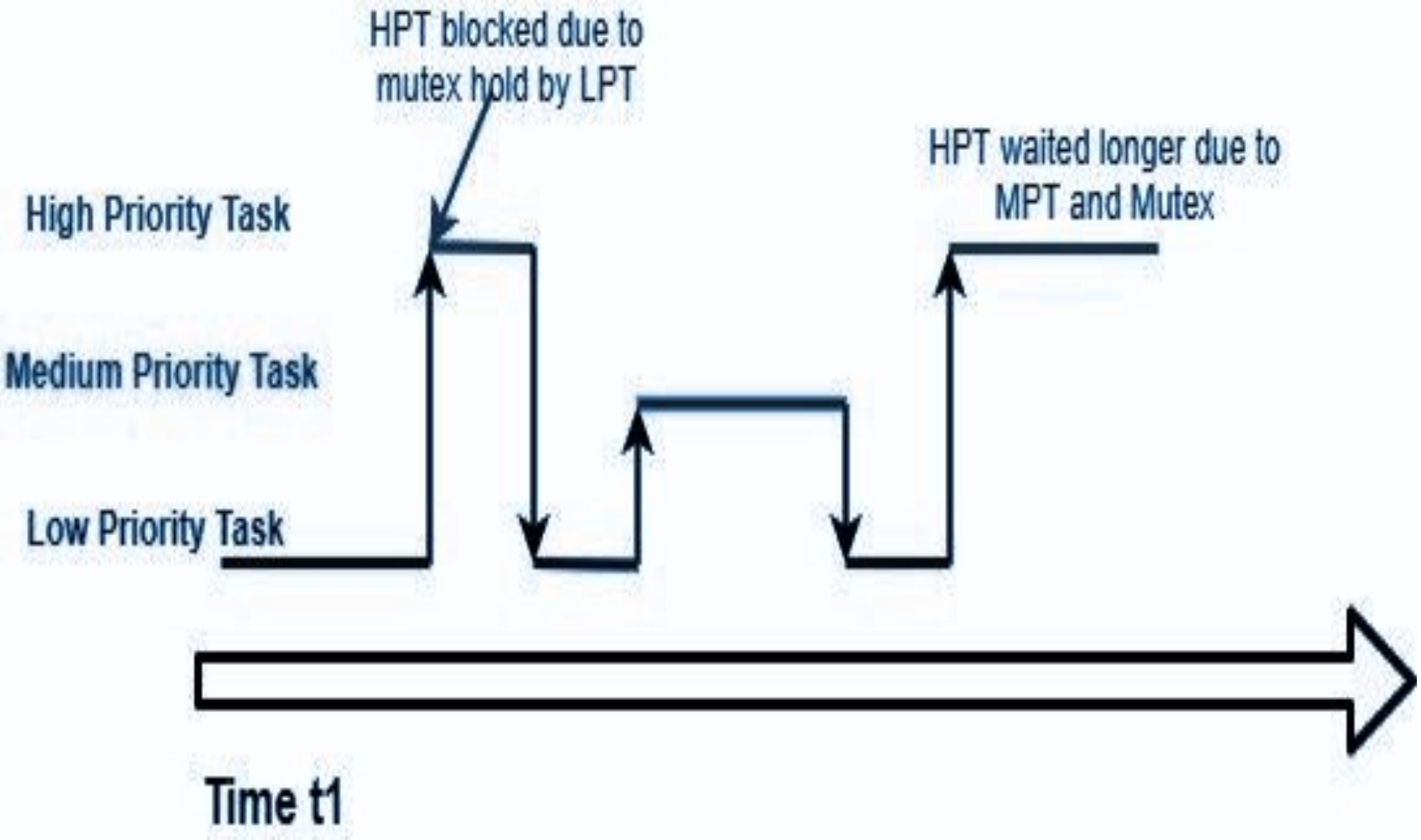


Assume that priorities: p1>p2>p3, and tasks 1 and 3 share the same critical resource

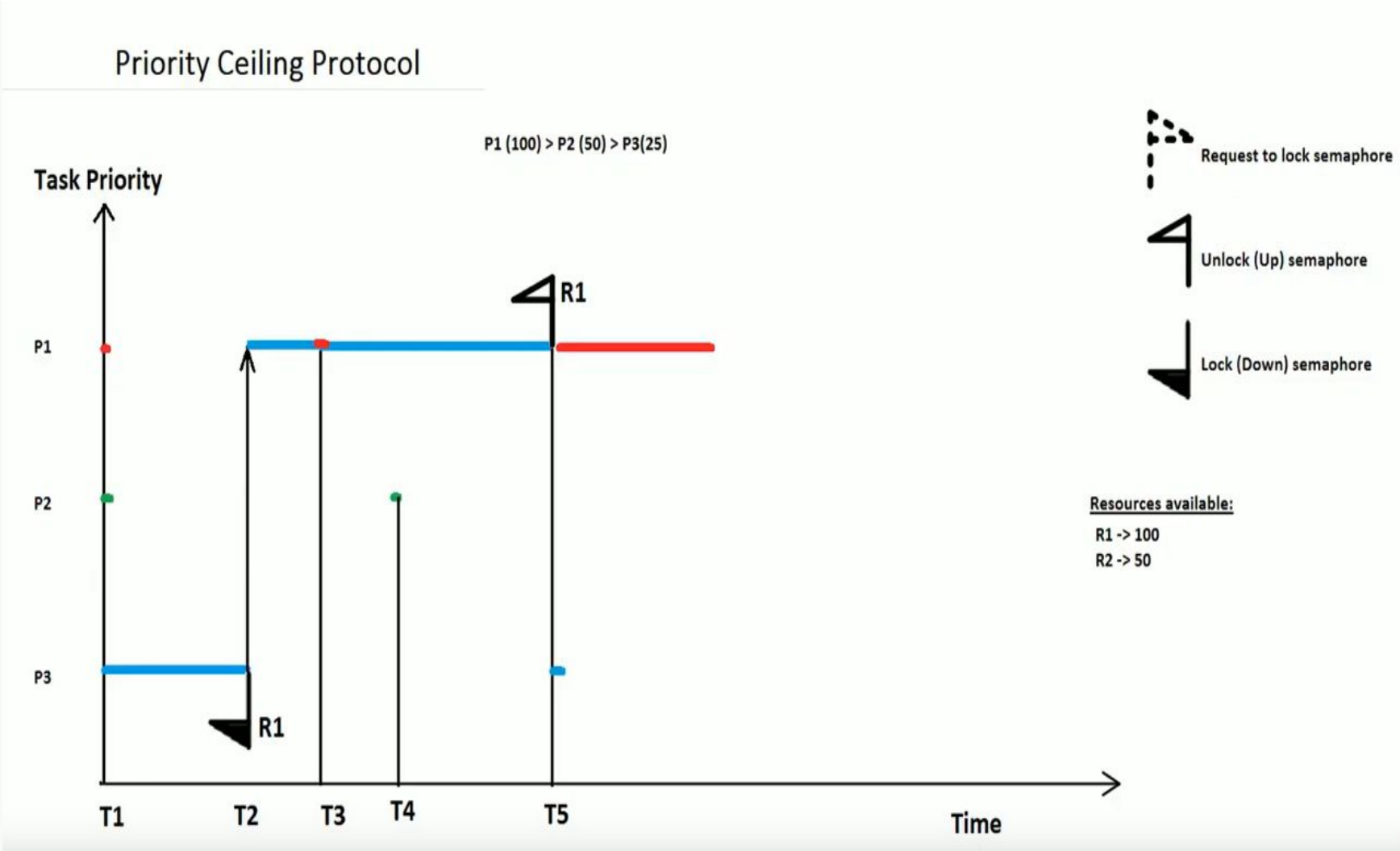
Priority Inversion versus Priority Inheritance



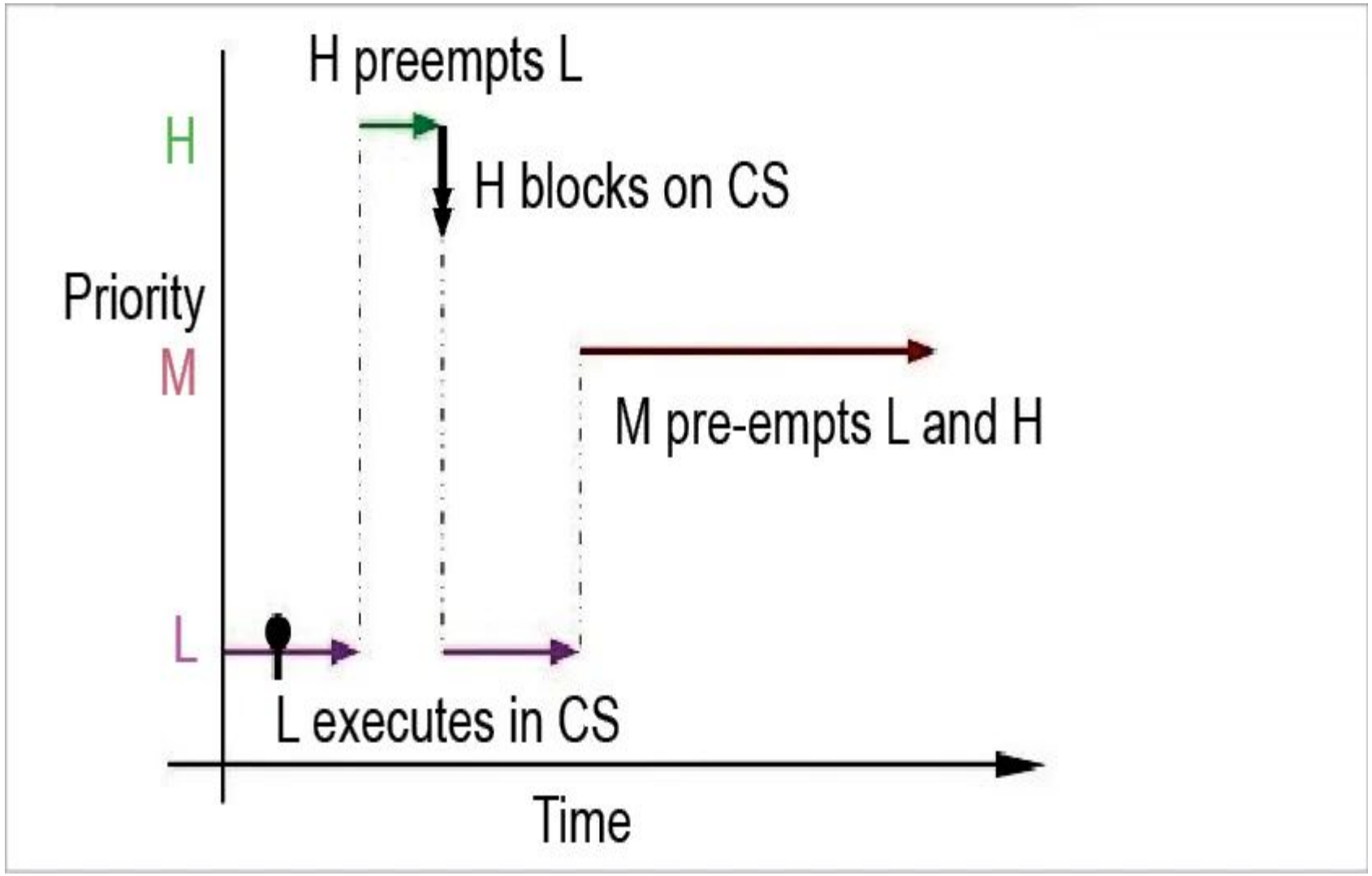
Priority Inversion versus Priority Inheritance



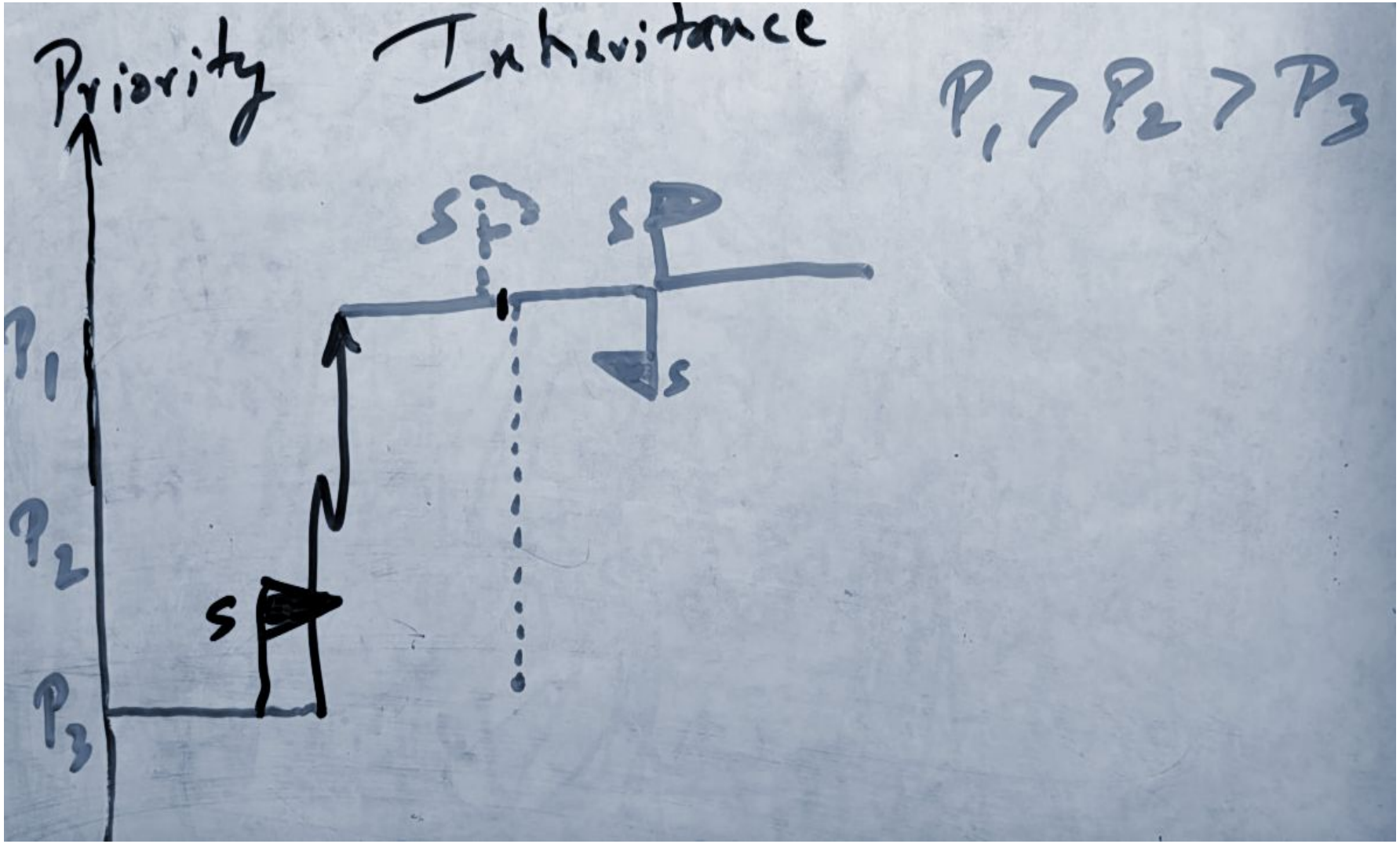
Priority Inversion versus Priority Inheritance



Priority Inversion versus Priority Inheritance



Priority Inversion versus Priority Inheritance



OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What common events lead to the creation of a process?**
  - A new batch job.
  - User performs interactive logon.
  - OS creates a process to provide a service.
  - An existing process spawns a new process

- **What does it mean to preempt a process?**
  - Process preemption occurs when an executing process is interrupted by the processor so that another process can be executed.

- **Why are two modes (user and kernel) needed?**
  - The user mode has restrictions on the instructions that can be executed and the memory areas that can be accessed. This is to protect the operating system from damage or alteration. In kernel mode, the operating system does not have these restrictions, so that it can perform its tasks.

- **What are the steps performed by an OS to create a new process?**
  - Assign a unique process identifier to the new process.
  - Allocate memory space for the process.
  - Initialize the process control block.
  - Set the appropriate linkages (to things like memory page tables, file handle tables, etc).
  - Create or expand other data structures as needed.

Can you answer the following typical questions related to Operating Systems ?

Why there should be separation between OS policy and OS mechanism ?

- The policies what is to be done while the mechanism specifies how it is to be done. For instance, the timer construct for ensuring CPU protection is mechanism. On the other hand, the decision of how long the timer is set for a particular user is a policy decision.

- The separation of mechanism and policy is important to provide flexibility to a system. If the interface between mechanism and policy is well defined, the change of policy may affect only a few parameters. On the other hand, if interface between these two is vague or not well defined, it might involve much deeper change to the system.

- Once the policy has been decided it gives the programmer the choice of using his/her own implementation. Also, the underlying implementation may be changed for a more efficient one without much trouble if the mechanism and policy are well defined. Specifically, separating these two provides flexibility in a variety of ways. First, the same mechanism can be used to implement a variety of policies, so changing the policy might not require the development of a new mechanism, but just a change in parameters for that mechanism, but just a change in parameters for that mechanism from a library of mechanisms. Second, the mechanism can be changed for example, to increase its efficiency or to move to a new platform, without changing the overall policy.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What is the difference between a mode switch and a process switch?**
  - A mode switch may occur without changing the state of the process that is currently in the running state. A process switch involves taking the currently executing process out of the running state in favor of another process. This process switch involves saving more state information.

- **Is it possible that you would want to allow a process to wait on more than one event at time. Explain with an example?**
  - An application may be processing data received from another process and storing the results on disk.
  - If there is data waiting to be taken from the other process, the application may proceed to get that data and process it. If a previous disk write has completed and there is processed data to write out, the application may proceed to write to disk. There may be a point where the process is waiting both for additional data from the input process and for disk availability.

- **In a number of early computers , an interrupt caused the register values to be stored in fixed locations associated with the given interrupt signal. Under what circumstances is this a practical technique?**

  - This technique is based on the assumption than an interrupted process A will continue to run after the response to an interrupt , and can immediately have its registers restored by the hardware. But, in general, an interrupt may cause the basic monitor to preempt a process A in favor of another process B. It is now necessary to copy the execution state of process A from the location associated with the interrupt to the process description associated with A. The machine might as well have stored them there in the first place.

Can you answer the following typical questions related to Operating Systems ?

- **In the case that you would want to allow a process to wait more than one event at a time, how would you modify the queuing structure to support this new feature?**
  - There are several ways that could be handled. A special type of either/or queue could be used. Or the process could be put in two separate queues. In either case, the operating system would have to handle the details of alerting the process to the occurrence of both events, one after the other.

- **Consider an environment in which there is a one-to-one mapping between user-level threads and kernel-level threads (KLT) that allows one or more threads within a process to issue system blocking calls while other threads continue to run. Explain why this model can make multi-threaded programs run faster than their single-threaded counterparts on uni-processor computers.**
  - The issue here is that a machine spends a considerable amount of its waking hours waiting for I/O to complete. In a multithreaded program, one KLT can make the blocking system call, while the other KLTs can continue to run. On uniprocessors, a process that would otherwise have to block for all these calls can continue to run its other threads.

- **Is busy waiting always less efficient (in terms of using processor time) than a blocking wait? Explain.**
  - On average, busy-waiting is more costly in terms of wasted CPU cycles. However, if the event condition is going to be satisfied within the next few machine cycles, it will be costly to incur the overhead of the OS dispatcher removing the process (for a block) and replacing it with another. The dispatcher is code executing in the OS and it uses up CPU cycles to do its tasks, too.

Can you answer the following typical questions related to Operating Systems ?

- **How are multiple interrupts dealt with?**
  - There are two approaches. The first is to disable interrupts while an interrupt is being processed. A disabled interrupt simply means that the processor ignores any new interrupt request signal. Any interrupt that occurs during this time will remain pending until the processor has re-enabled interrupts.
  - The second approach is to define priorities for interrupts and allow interrupts of higher priority to cause a lower-priority interrupt handler to be interrupted.

- **What characteristics distinguish the various elements of a memory hierarchy?**
  - Decreasing cost per bit.
  - Increasing capacity.
  - Increasing access time.
  - Decreasing frequency of access to the memory by the processor.

- **What is the distinction between spatial locality and temporal locality?**
  - Temporal locality is the concept that a resource that is referenced at one point in time will be referenced again sometime in the near future.
  - Spatial locality is the concept that the likelihood of referencing a resource is higher if a resource near it was just referenced.

Can you answer the following typical questions related to Operating Systems ?

- **In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access to main memory. Why?**
  - When transferring a large chunk of data, it is more efficient to use DMA. It is designed to do laborious data transfers instead of slowing down performance by wasting CPU intelligence on the task

- **What is a process ?**
  - A program in execution.
  - An instance of a program running on a computer.
  - The entity that can be assigned to and executed on a processor.
  - A unit of activity characterized by the execution of a sequence of instruction a current state, and an associated set of system resources.

Can you answer the following typical questions related to Operating Systems ?

- **Three examples of an interrupt?**
  - Clock interrupt - The OS determines whether the currently running process has been executing for the maximum allowable unit of time. If so, this process must be switched to a Ready state and another process dispatched.
  - I/O Interrupt - The OS determines what I/O Interrupt has occurred. If the I/O Interrupt constitutes an event for which one or more processes are waiting then the OS moves all of the corresponding blocked processes to the Ready state.
  - Memory fault - The processor encounters a virtual memory address reference for a word that is not in main memory. The OS must bring in the block of memory containing the reference from secondary memory to main memory.

- **What is swapping and what is its purpose?**
  - Involves moving part or all of a process from main memory to disk. The purpose is to increase the utilization of the processor.

Can you answer the following typical questions related to Operating Systems ?

- **List four characteristics of a suspended process.**
  - 1. the process is not immediately available for execution.
  - 2. The process may or may not be waiting on an event. if it is, this blocked condition is independent of the suspend condition, and occurrence of the blocking event does not enable to process to be executed immediately.
  - 3. The process is placed in a suspended state by an agent: either itself or a parent process, or the OS, for the purpose of preventing its execution.
  - 4. The process may not be removed from this state until the agent explicitly orders the removal.
- **What is the difference between an interrupt and a trap?**
  - Interrupts are hardware interrupts.
  - Traps are software interrupts.

Can you answer the following typical questions related to Operating Systems ?

- **What are the two separate and potentially independent characteristics embodied in the concept of process?**
  - Resource ownership and scheduling/execution.

- **Give four general examples of the use of threads in a single-user multiprocessing system.**
  - Foreground and background work.
  - Asynchronous processing.
  - Speed of execution.
  - Modular program structure.

- **List four design issues for which the concept of concurrency is relevant.**
  - 1. Communication among processes.
  - 2. Sharing of and competing for resources.
  - 3. Synchronization of the activities of multiple processes.
  - 4. Allocation of processor time to processes.

Can you answer the following typical questions related to Operating Systems ?

- **List three degrees of awareness between processes and briefly define each.**
  - Processes unaware of each other - In this case OS needs to be concerned about competition for resources.
  - Processes indirectly aware of each other - Processes exhibit cooperation sharing a common object even if they aren't aware of each other.
  - Processes directly aware of each other - processes share common object but are designed to work jointly on some activity.
- **What is the distinction between competing processes and cooperating processes?**
  - Cooperation is sharing an object instead of claiming it all.

Can you answer the following typical questions related to Operating Systems ?

- **List the requirements for mutual exclusion.**
  - 1. Mutual exclusion must be enforced: Only one process at a time is allowed into its critical section, among all processes that have critical section for the same resource or shared object.
  - 2. A process that halts in its non critical section must do so without interfering with other processes.
  - 3. It must not be possible for a process requiring access to a critical section to be delayed indefinitely: no deadlock or starvation.
  - 4. When no process is in a critical section, any process that requests entry to its critical section must be permitted to enter without delay.
  - 5. No assumptions are made about relative process speeds or number of processors.
  - 6. A process remains inside its critical section for a finite time only.

Can you answer the following typical questions related to Operating Systems ?

- **What operations can be performed on a semaphore?**
  - 1. Can be initialized to a non-negative integer value.
  - 2. The wait operation decrements the semaphore value. if the value becomes negative then the process executing the wait is blocked. Otherwise the process continues execution.
  - 3. The signal operation increments the semaphore value. If the resulting value is less than or equal to zero, then a process blocked by a wait operation, if any, is unblocked.

Can you answer the following typical questions related to Operating Systems ?

- **What is the difference between binary and general semaphores?**
  - A binary semaphore may only take on the values 0 and 1,
  - The wait operation checks the semaphore value. If the value is zero, then the process executing the wait is blocked. If the values is one, then the value is changed to zero and the process continues execution.
  - The signal operation checks to see if any processes are blocked on the semaphore. If so, then a process blocked by a wait operation is unblocked. if no processes are blocked, then the value of the semaphore is set to one.

Can you answer the following typical questions related to Operating Systems ?

- **What is the difference between strong and weak semaphores?**
  - A semaphore that does not specify the order in which processes are removed from the queue is considered a weak semaphore.
  - A semaphore whose definition includes the policy of the order of its removal from queue is called a strong semaphore.

Can you answer the following typical questions related to Operating Systems ?

- **What conditions are generally associated with the readers/writers problem?**
  - 1. Any number of readers may simultaneously read the file.
  - 2. Only one writer at a time may write to the file.
  - 3. If a writer is writing to the file, no reader may read it.

Can you answer the following typical questions related to Operating Systems ?

- **A computer has a cache, main memory, and a disk used for virtual memory. If a referenced word is in the cache, 20 ns are required to access it. If it is in main memory but not in the cache, 60 ns are needed to load it into the cache (this includes the time to originally check the cache), and then the reference is started again. If the word is not in main memory, 12 ms are required to fetch the word from disk, followed by 60 ns to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main-memory hit ratio is 0.6. What is the average time in ns required to access a referenced word on this system?**

- Assuming a reference is in memory or cache we get an average access time of 0.9 ×20 ns + 0.1×80 ns = 26 ns

- Now considering the possibility of going to disk, we get
  0.99×26 ns + 0.01×10000080 ns = 100,026.54 ns ~= 100µs

- Clearly we need very high hit ratios to make a virtual memory system work well.

- An alternative interpretation is that the .99 main memory hit ratio refers to the 0.99 of the .1 requests that go to main memory.
- In that case we have
  0.9×20 ns + 0.1(.99×80 ns +.01×10000080 ns) = 10,026.ns = 10µs

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Explain the difference between a monolithic kernel and a microkernel.**

- Monolithic kernel: A large kernel containing virtually the complete operating system, including scheduling, file system, device drivers, and memory management. All the functional components of the kernel have access to all of its internal data structures and routines. Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space.

- Microkernel: A small privileged operating system core that provides process scheduling, memory management, and communication services and relies on other processes to perform some of the functions traditionally associated with the operating system kernel.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- Periodic tasks => Are completed regularly, once per period T or T units apart
- Aperiodic tasks =>Have time constraints either for deadlines or start
- Rate Monotonic Scheduling
  - Assigns priorities to tasks on the basis of their periods
  - Highest-priority task is the one with the shortest period
  - The second highest-priority task is the one with the second shortest period
- **Why is it not possible to enforce memory protection at compile time?**
  - The OS cannot anticipate all the memory references a program will make, and even if it could, it would be prohibitively time consuming to screen each program in advance for possible memory-reference violations.
- **What are some reasons to allow two or more processes to all have access to a particular region of memory?**
  - If a number of processes are executing the same program, it is advantageous to allow each process to access the same copy of the program rather than have its own copy. Processes that are cooperating on some task may need to share access to the same data structure.
-

Can you answer the following typical questions related to Operating Systems ?

- **In a fixed-partitioning scheme, what are the advantages of using unequal-size partitions?**
  - Processes are assigned in such a way as to minimize wasted memory within a partition (internal fragmentation). Larger programs can be accommodated without overlay.

- **What are the distinctions among logical, relative, and physical addresses?**
  - A logical address is a reference to a memory location independent of the current assignment of data to memory; a translation must be made to a physical address before the memory access can be achieved.
  - A relative address is a particular example of logical address, in which the address is expressed as a location relative to some known point, usually a value in a processor register.
  - A physical address is an actual location in main memory.

- **What is the difference between simple paging and virtual memory paging?**
  - In contrast to simple paging, not all pages of a process have to be in main memory for the process to run. Pages may be read in as needed. Also, in virtual memory paging, reading a page into main memory may require writing a page out to disk.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **Distinguish between blocking and nonblocking with respect to messages**
  - Both sender/receiver can be blocking or nonblocking
  - 3 combinations
    - 1. Blocking send / receive-: both sender and receiver are blocked until message is delivered
    - 2. Non-blocking send / blocking receive-: sender may continue on -receiver is blocked until message arrives
    - 3. Non-blocking send / receive -: neither party waits

- **What conditions are generally associated with the readers/writers problem**
  - Any number of readers may simultaneously read
  - Only one writer at a time may write
  - If writing, no reader may read
  - readers are processes that are not required to exclude one another
  - writers are processes that are required to exclude all other processes -readers and writers.
  - readers and writers may have priorities
  - this is producer/consumer relationship can't produce if full -can't consume if empty

Can you answer the following typical questions related to Operating Systems ?

- **Give examples of reusable and consumable resources**
  - Reusable resources include processors; I/O Channels; main and secondary memory; devices; etc.
  - Consumable resources include interrupts; signals; messages and buffers

- **What are the three typical General Approaches for Deadlock ?**
  - 1. Prevent
  - 2. Avoid
  - 3. Detect
  - Mutual Exclusion cannot be disallowed
  - fatal region -process enters, deadlock is inevitable

- **How can the hold-and-wait condition be prevented ?**
  - Can be prevented by requiring that a process request all of its required resources at one time
  - blocking until all requests can be granted simultaneously

Can you answer the following typical questions related to Operating Systems ?

- **What is the difference among deadlock Avoidance, Detection, and Prevention ?**
  - Prevention attempts to constrain resource requests to prevent at least one of the four conditions
  - Avoidance allows 3 of the conditions but makes choices aimed at assuring deadlock point isn't reached
  - Detection performs resource checks to try and detect deadlock condition so they can be avoided
- **What requirements is memory management intended to satisfy ?**
  - 1. Relocation
  - 2. Protection
  - 3. Sharing
  - 4. Logical Organization
  - 5. Physical Organization

Can you answer the following typical questions related to Operating Systems ?

- **Why is the capability to relocate processes desirable ?**
  - When a program has been swapped out to disk it would be quite limiting to declare that when swapped back in - it be placed in the same location
  - Relocating the process allows processes to be in different places within memory
  - Relocation - moving a process to different memory area
- **What is the difference between simple paging and virtual memory paging ?**
  - Simple paging requires all pages of the process be loaded into main memory
  - Virtual memory allows pages to be read into main memory as needed
  - Virtual memory may require pages to be written out to disk
  - Virtual memory improves performance by allowing more processes resident
  - Virtual memory allows a process to be larger than all of main memory

Can you answer the following typical questions related to Operating Systems ?

- **What is concept behind deadlock avoidance ?**
    - A decision is made dynamically whether the current resource allocation request will lead to deadlock. This requires a knowledge of future process requests
    - Two approaches:
        - 1. process initiation denial - don’t start a process if its demands may lead to deadlock. this assumes the worst - that all processes will make their max claims together
        - 2. Resource allocation denial - dont grant an incremental resource request to a process if it may lead to a deadlock (bankers algorithm). when a process makes a request for a set of resources, assume it is granted and update the system accordingly. Determine if the result is a safe state, and if it is, grant the request
- **What is the concept behind deadlock detection ?**
    - Resource requests are granted whenever possible,
    - Regularly check for deadlock
    - Once deadlock is detected, abort all deadlocked processes, back up each deadlocked process to some previously defined checkpoint and restart all processes, successively aborting deadlocked processes until deadlock no longer exists

Can you answer the following typical questions related to Operating Systems ?

- **What is the Logical organisation of program ?**
    - Memory is usually organized linearly. Programs are written in modules which can be written and compile independently. Different degrees of protection are given to modules. Segmentation helps here

- **Explain the terms**

- **Physical Organization**
    - Cannot leave the programmer with the responsibility to manage main memory, and the programmer does not know how much space will be available. Memory available for a program plus its data may be insufficient
- **Fixed Partitioning**
    - Equal size partitions, and any process whose size is less than or equal to the partition size can be loaded into an available partition. The OS can swap a process out of a partition if none are in a ready or running state. Problems with this method are that a program may not fit in a partition, and main memory use is inefficient (internal fragmentation)

- **Unequal Size Fixed Partitioning**
    - Partitions are not equally sized. we can assign each process to the smallest partition within which it will fit, and there is a queue for each partition. The problem is still that a large number of very small processes will not use the space efficiently

Can you answer the following typical questions related to Operating Systems ?

- **Explain the terms**

- **Dynamic Partitioning**
  - Partitions are of variable length and number and process is allocated exactly as much memory as required. Memory external to all processes is fragmented - external fragmentation. this can be resolved using compaction (OS moves processes so that they are contiguous. this is time consuming and wastes CPU time). there are multiple algorithms for determining the best fit for a block

- **What happens when an address that is not needed is in main memory ?**
  - An interrupt is generated, the OS places the process in a blocking state. Because of this, more processes may be maintained in main memory and a process may be larger than all of main memory

- **What is Fetch Policy ?**
  - Determines when a page should be brought into memory. demand paging only brings pages into main memory when a reference is made to a location on the page, which results in many page faults when process first started prepaging brings in more pages than needed, it is more efficient to bring in pages that reside contiguously on the disk

Can you answer the following typical questions related to Operating Systems ?

- **What is Page Placement Policy ?**
  - Determines where in real memory a process piece is to reside

- **What is Page Replacement Policy ?**
  - Determines which page currently in memory is to be replaced when all frames are occupied and it is necessary to bring in a new page.

  - Basic replacement algorithms include optimal (selects for replacement the page for which the time to the next reference is the longest), least recently used (replaces the page that has not been referenced for the longest time), first in first out (page that has been in memory the longest is replaced), and clock (when a page is first loaded in memory or referenced, the use bit is set to 1. when it is time to replace a page, the OS scans the set, flipping all 1's to 0. The first frame encountered with a 0 already is replaced)

- **What is Page Cleaning Policy ?**
  - Concerned with determining when a modified page should be written out to secondary memory.

  - Demand cleaning is where a page is written out only when it has been selected for replacement precleaning is where pages are written out in batches the best approach uses page buffering- replaced pages are placed in two lists, modified (pages in this list are periodically written out in batches) and unmodified (pages are reclaimed if referenced again, or lost when its frame is assigned to another page)

Can you answer the following typical questions related to Operating Systems ?

- **What do you mean by Load Control ?**
  - Determines the number of processes that will be resident in main memory - too few processes lead to many occasions when all processes will be blocked and much time will be spent in swapping, but too many processes will lead to thrashing

- **What is the process of Process Suspension ?**
  - If the degree of multiprogramming is to be reduced, one or more of the currently resident processes must be suspended.

  - This can be decided by the **lowest priority process**, **faulting process** (process does not have its working set in main memory, so it will be blocked anyway), **last process** activated (this process is least likely to have its working set resident), process with smallest resident set (process requires the least future effort to reload), **largest process** (obtains the most free frames) or the process with the largest remaining execution window

- **What is Predictability ?**

  - A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system turning to cure instabilities

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems ?

- **What do you mean by Load Control ?**
  - Determines the number of processes that will be resident in main memory - too few processes lead to many occasions when all processes will be blocked and much time will be spent in swapping, but too many processes will lead to thrashing

- **What is the process of Process Suspension ?**
  - If the degree of multiprogramming is to be reduced, one or more of the currently resident processes must be suspended.

  - This can be decided by the **lowest priority process**, **faulting process** (process does not have its working set in main memory, so it will be blocked anyway), **last process** activated (this process is least likely to have its working set resident), process with smallest resident set (process requires the least future effort to reload), **largest process** (obtains the most free frames) or the process with the largest remaining execution window

- **What is Predictability ?**

  - A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system turning to cure instabilities

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems => True / False ?

- For a fixed partition system, memory deallocation is relatively complex => **False**
- Onboard systems are computers that are physically placed inside the products that they operate to add features and capabilities. => **False**
- The content of a random access memory (RAM) chip is nonvolatile, meaning that it is not erased when the power is turned off.=> **False**
- With demand paging, if there are no empty page frames available, to move in a new page, one of the current resident pages must be placed into main memory. => **False**
- The fixed partition scheme works well if all of the jobs run on the system are of the same size or if the sizes are known ahead of time and don't vary between reconfigurations. => **True**
- In a dynamic partition system, a null entry in the busy list occurs when a memory block between two other busy memory blocks is returned to the free list => **True**
- Virtual memory can be implemented with both paging and segmentation. => **True**

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems => Fill in the Blanks ?

- In demand paging, pages are **Swapped** between main memory and secondary storage.
- In a paged memory allocation scheme, the **Job Table** contains two values for each active job: the size of the job and the memory location where its Page Map Table is stored.
- The **FIFO** page replacement policy is based on the theory that the best page to remove is the one that has been in memory the longest.
- A variation of the LRU page replacement algorithm known as the **Clock Page** replacement policy is implemented with a circular queue.
- There are two types of real-time systems depending on the consequences of missing the deadline. A **Hard** real-time system risks total system failure if the predicted time deadline is missed.
- The **Belady's Anomaly** demonstrates that when using a FIFO policy, in rare cases, adding more memory to a system can cause an increase in page interrupts.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems => Fill in the Blanks ?

- The **First Fit** method keeps the free/busy lists organized by memory locations, from low-order memory to high-order memory.

- The goal of the **Best Fit** memory allocation algorithm is to find the smallest memory block into which a job will fit

- A disadvantage of segmented memory allocation is **External Fragmentation**

- The purpose of cache memory is to keep handy the most recently accessed data and instructions so that the CPU can access them repeatedly without wasting time. This purpose is similar to that of the **Bookmark** of a Web browser.

- Deadlock can occur on a printer when **the printer needs all of a job's output before it will begin printing, but the spooling system fills the available disk space with only partially completed output**

- No movement between queues is a very simple policy that rewards those who have **High Priority** jobs.

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems => Fill in the Blanks ?

- The **Loosely Coupled** multiprocessing configuration features several complete computer systems, each with its own memory, I/O devices, CPU, and operating system.

- The Banker's Algorithm is typically an example of an **Deadlock Avoidance** policy

- A computer system that can support jobs that use multiple processors to execute sets of instructions in parallel is referred to as a **Concurrent processing** system.

- **Earliest Deadline First** is a dynamic-priority preemptive scheduling algorithm built to address the critical processing requirements of real-time systems and their pressing deadlines.

- An algorithm designed to detect starvation by tracking how long each job has been waiting for resources is using the concept of **Aging**

- Most current operating systems support the implementation of threads, or **Light weight processes**, which have become part of numerous application packages

OPERATING SYSTEMS

Can you answer the following typical questions related to Operating Systems => Fill in the Blanks ?

- The Process Scheduler assigns the CPU to execute the processes for those jobs placed on the **ready** queue by the Job Scheduler.

- The transition from **running to normal exit** can be initiated by the Process Scheduler or the Job Scheduler.

- When a compiler automatically detects instructions that can be performed in parallel, **implicit parallelism** is in place.

- The transition from **hold to ready** is initiated by the Job Scheduler according to some predefined policy. At this point, the availability of enough main memory and any requested devices is checked

- Linux terminal management conforms to **POSIX** standards, and it also supports pseudo-terminals.

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean, IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com