**Thread**

1. Why is a thread called Light Weight process? Explain the kernel-level thread architecture.

Ans:

Threads are sometimes called lightweight processes because they have their own stack but can access shared data. Because threads share the same address space as the process and other threads within the process, the operational cost of communication between the threads is low, which is an advantage.

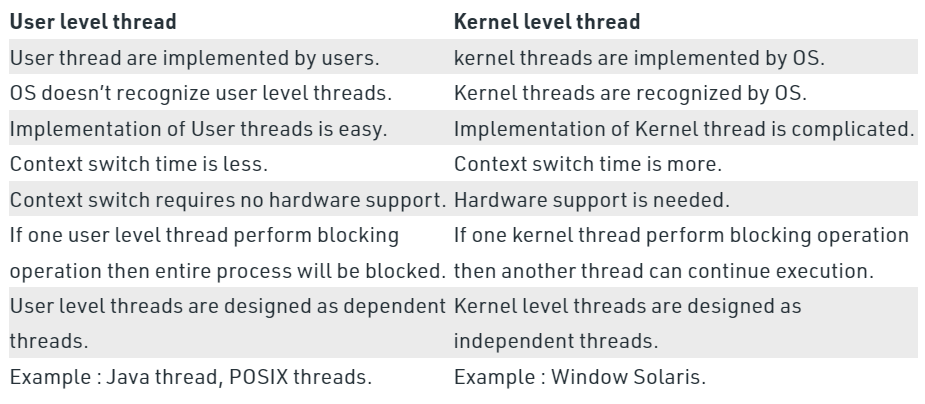
4_01.pdf

**Figure: Kernel level thread architecture**

Kernel-level threads are handled by the operating system directly and the thread management is done by the kernel. The context information for the process as well as the process threads is all managed by the kernel. Because of this, kernel-level threads are slower than user-level threads.

1. Difference between User level thread and Kernel level thread?

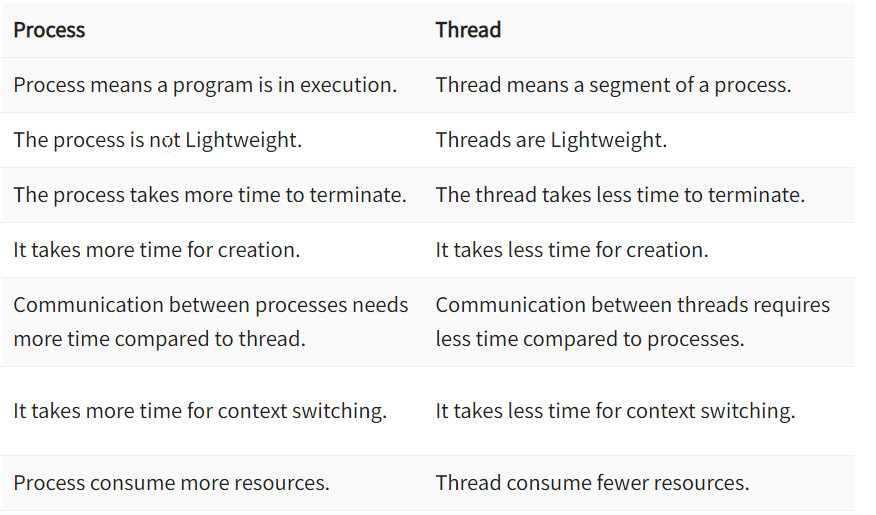
**Ans:**

****

1. When should we create thread instead of duplicating a process?

**Ans:**

* downloading a video while playing it at the same time
* Formatting a MS-Word file while typing at the same time



1. Multithreading Models

**One to One:**

4_06.pdf

**Many to One:**

4_05.pdf

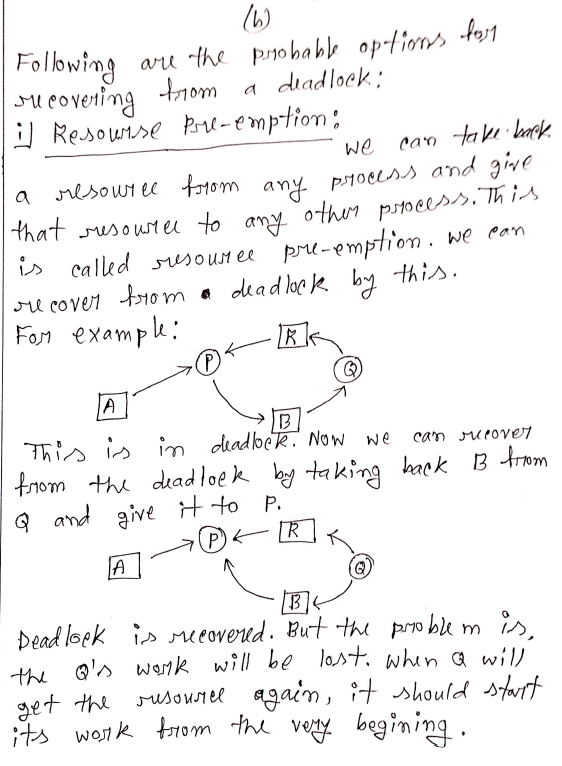
**Many to Many:**

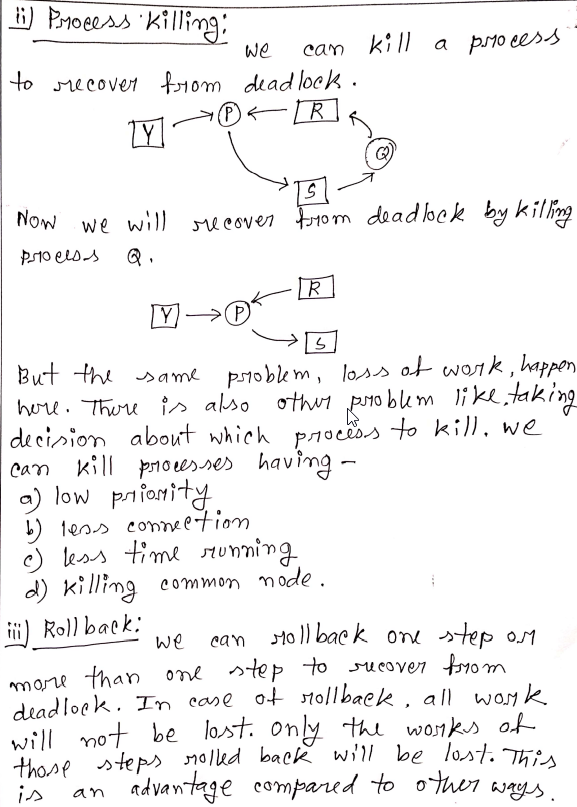
4_07.pdf

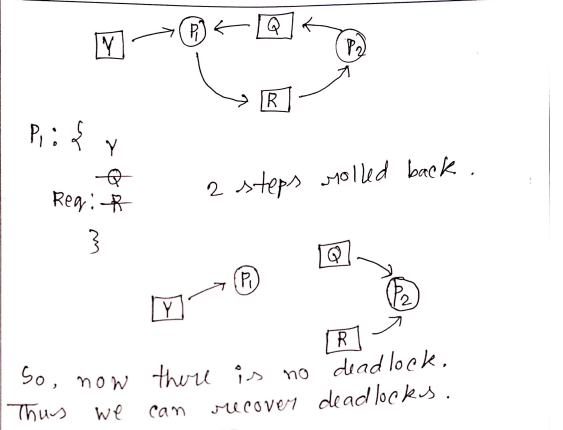
**Deadlock**

1. What are the probable options for **recovering** from a deadlock?

**Ans:**

****

****

****

1. Explain the causes where a set of processes can be trapped into a deadlock situation. How can a deadlock be **prevented**?

**Ans:**

**Deadlockis a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.**

Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



**Deadlock can arise if**the**following four conditions hold simultaneously (Necessary Conditions)**  
**Mutual Exclusion:** Two or more resources are non-shareable (Only one process can use at a time)   
**Hold and Wait:**A process is holding at least one resource and waiting for resources.   
**No Preemption:** A resource cannot be taken from a process unless the process releases the resource.   
**Circular Wait:** A set of processes are waiting for each other in circular form.

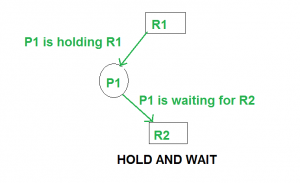
**Deadlock Prevention**

We can prevent Deadlock by eliminating any of the above four conditions.

**Eliminate Mutual Exclusion**  
It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

**Eliminate Hold and wait**

1. Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. for example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
2. The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

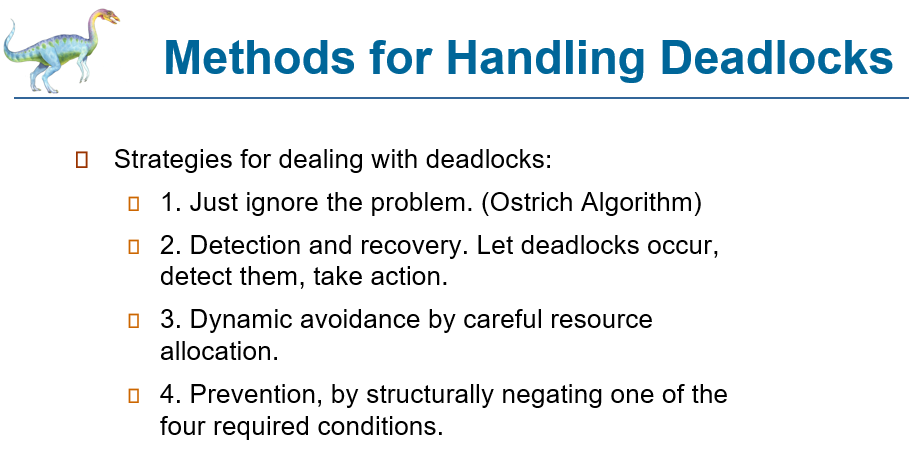


**Eliminate No Preemption**   
Preempt resources from the process when resources are required by other high-priority processes.

**Eliminate Circular Wait**   
Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing. order of numbering.   
For Example, if the P1 process is allocated R5 resources, now next time if P1 asks for R4, R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

1. What are the ways an Operating System has to deal with deadlocks in the system?

**Ans:**

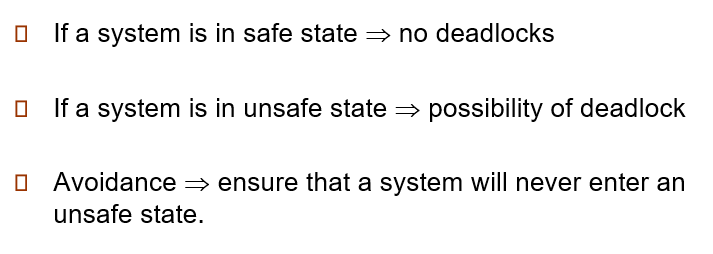
****

1. What is meant by a safe state?

Ans:

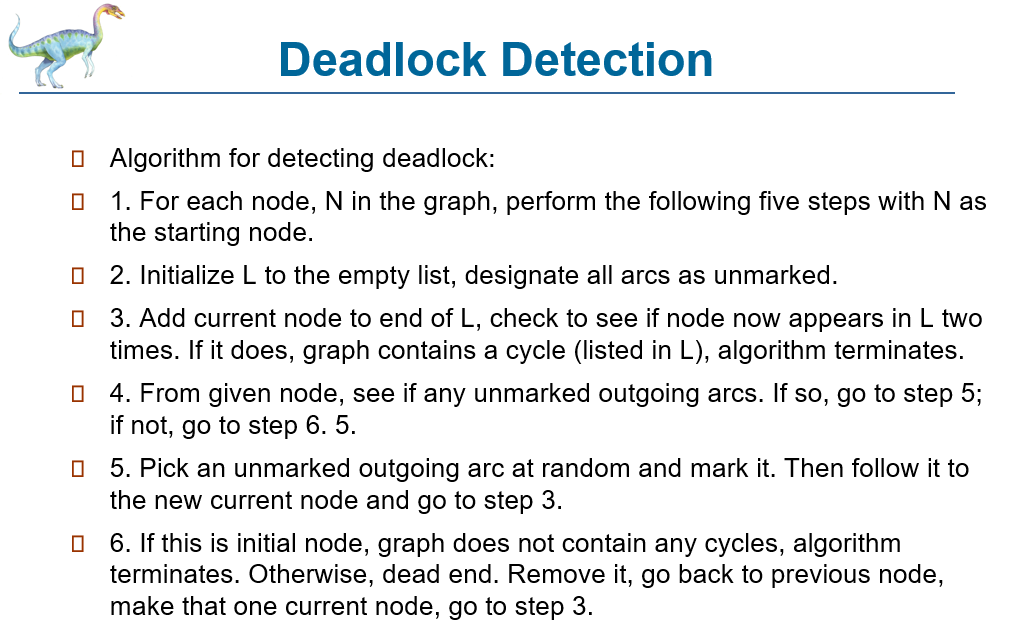
A state of the system is called safe if the system can allocate all the resources requested by all the processes without entering into deadlock.

If the system cannot fulfill the request of all processes, then the state of the system is called unsafe.



1. Show the Deadlock Detection Mechanism. (Algorithm)

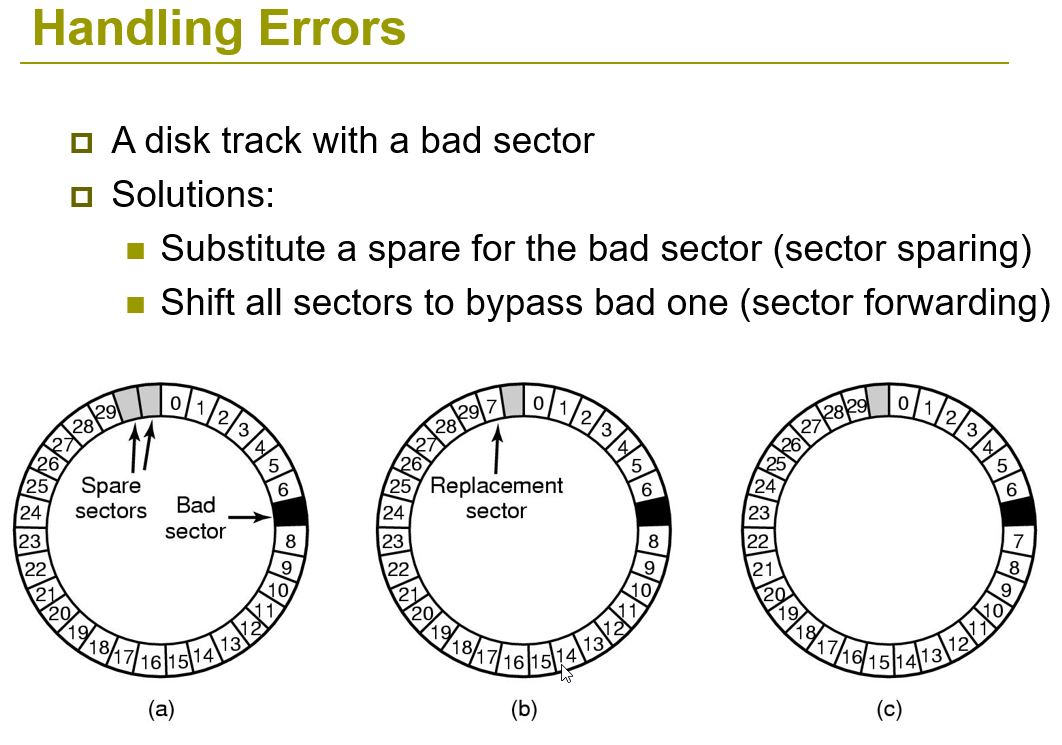
Ans:



**Mass Storage Management**

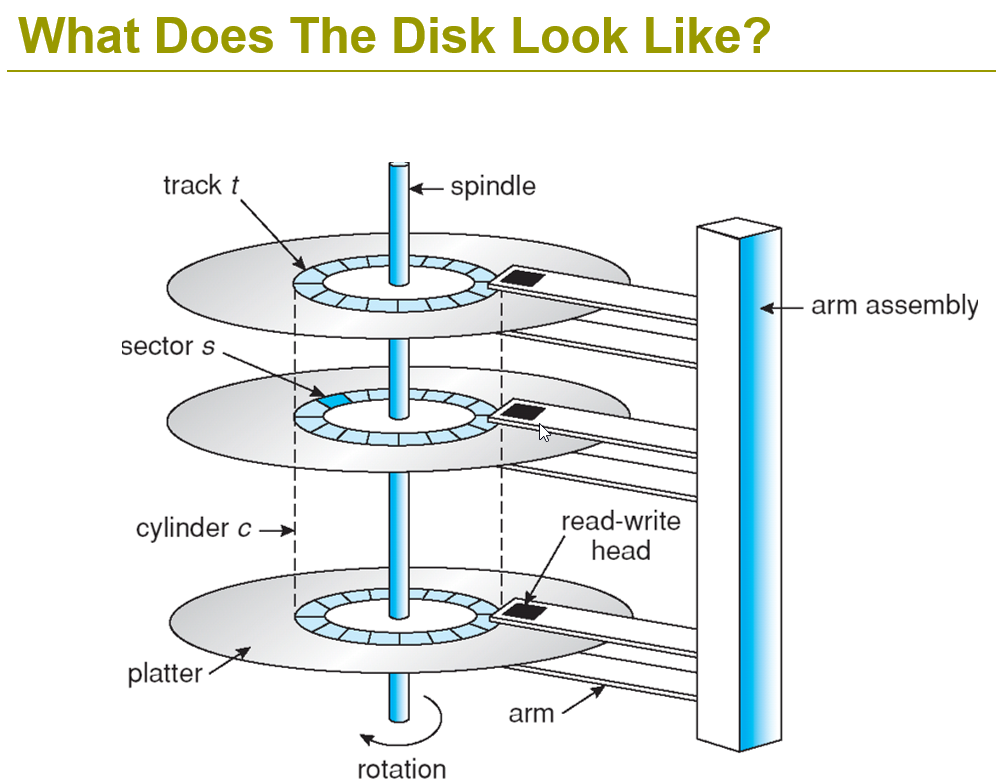
1. How can an OS handle a bad sector in a disk drive?

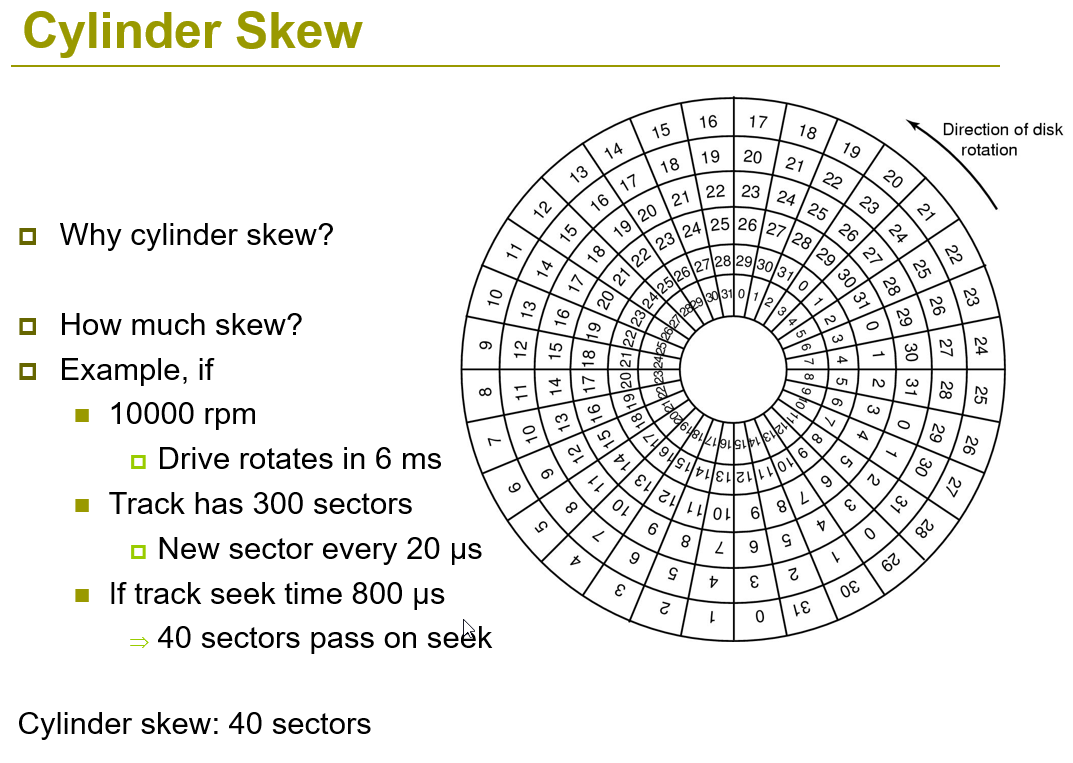
**Ans:**

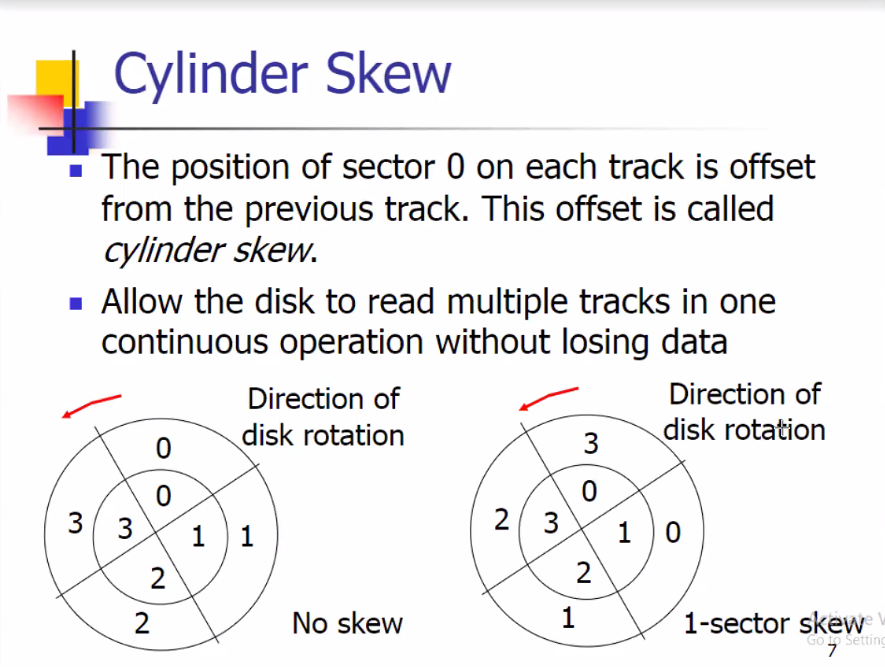
****

1. **What do you understand by cylinder skew in disk management system?**

**Ans:**

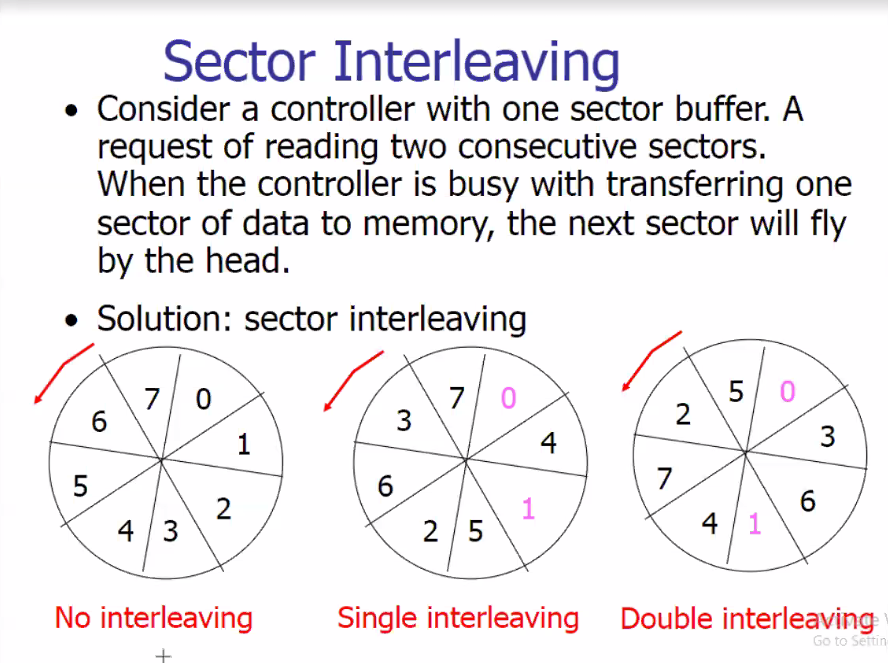
****

****



1. **What do you understand by sector interleaving?**

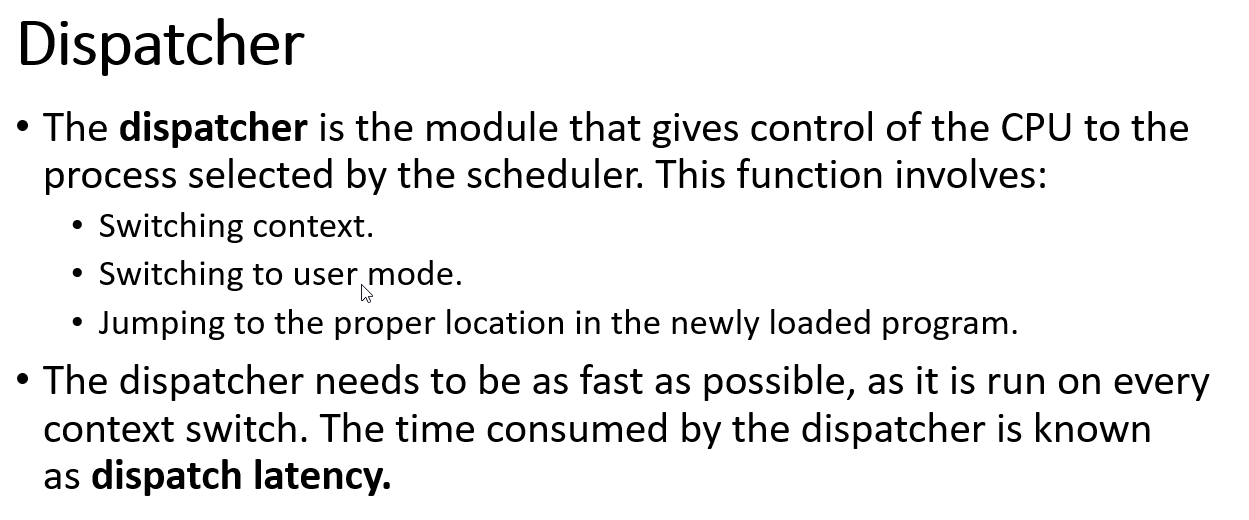
**Ans:**



**Process Scheduling**

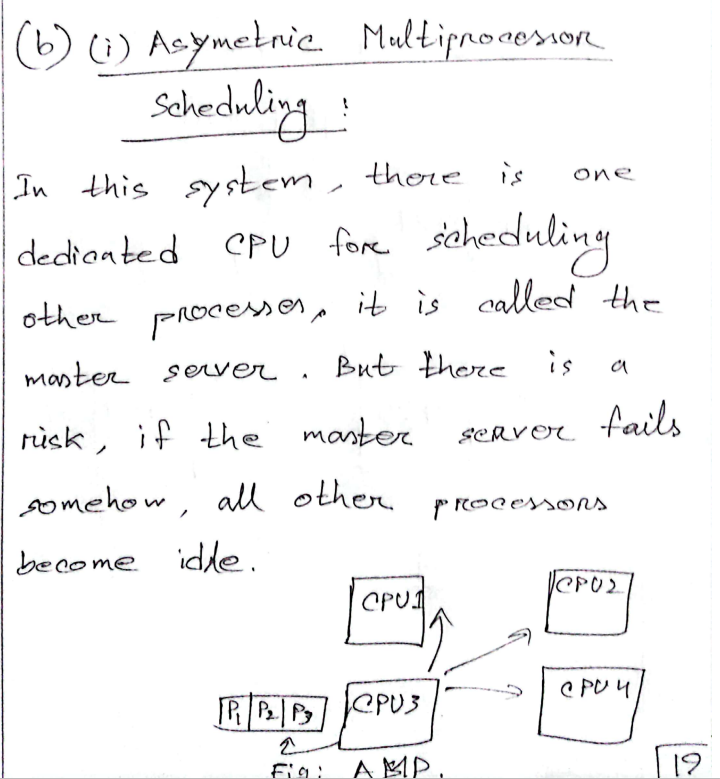
1. What is the function of the dispatcher module of an Operating System?

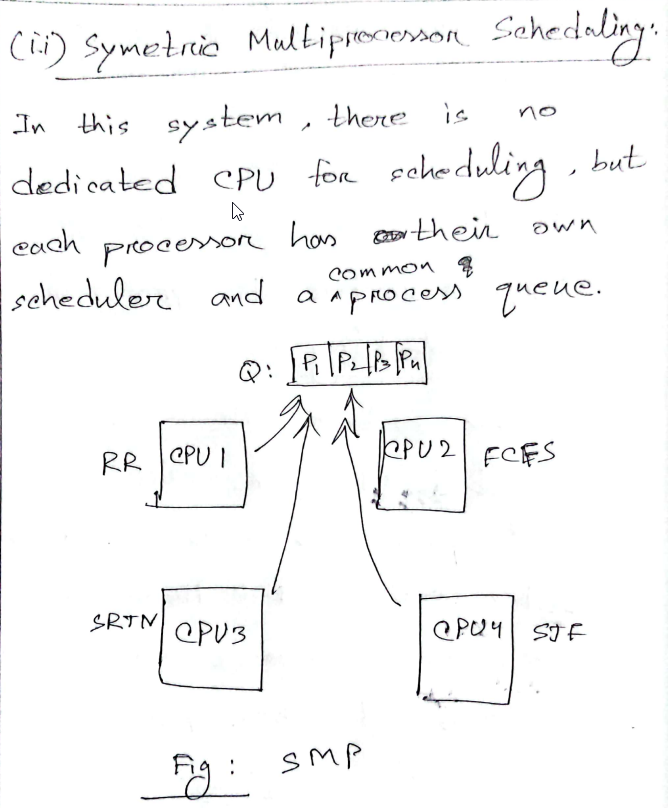
**Ans:**

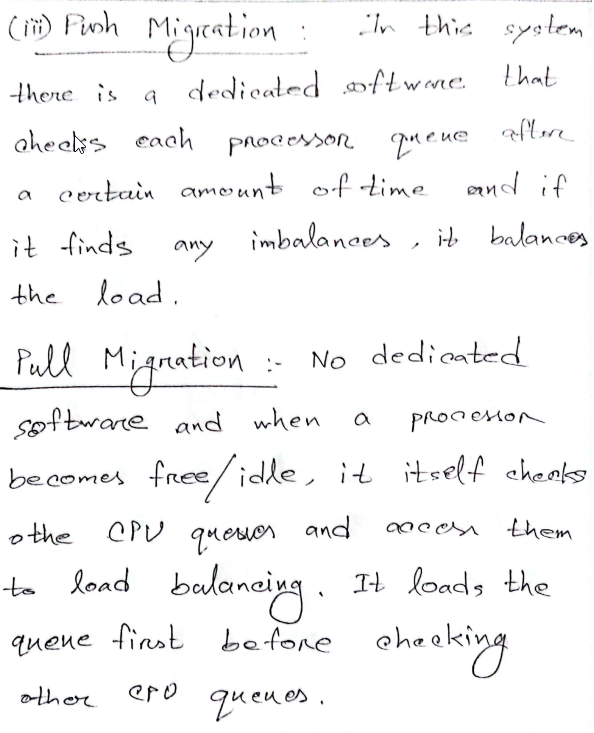
****

1. Explain the following terms with necessary figure(s) with respect to a multiprocessor-based scheduling system.
2. Asymmetric multiprocessor scheduling
3. Symmetric multiprocessor scheduling
4. Pull and Push migration during load balancing

Ans:







Balancing can be achieved through either ***push migration***or***pull migration:***

* ***Push migration***involves a separate process that runs periodically, (e.g., every 200 milliseconds), and moves processes from heavily loaded processors onto less loaded ones.
* ***Pull migration*** involves idle processors taking processes from the ready queues of other processors.

SMP systems attempt to keep processes on the same processor, via ***processor affinity.***

***Soft affinity*** occurs when the system attempts to keep processes on the same processor but makes no guarantees.

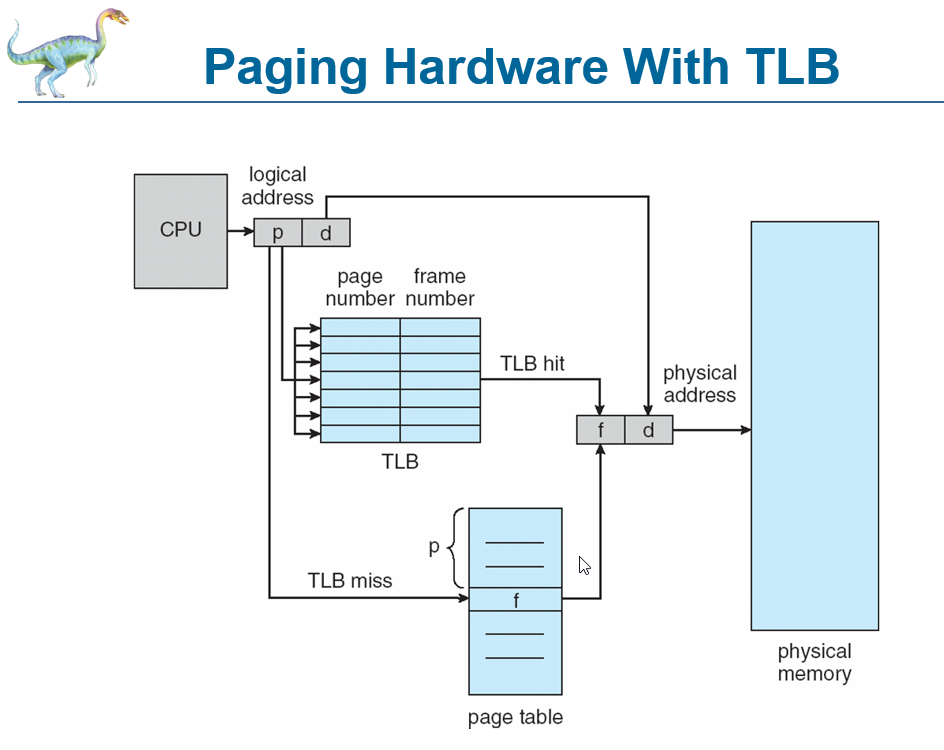
Linux and some other OSes support ***hard affinity,***in which a process specifies that it is not to be moved between processors.

**Memory Management**

1. Draw the diagram for an address translation process in any virtual memory system with a Translation Look-aside Buffer.
2. Discuss the fundamental concept of using Translation lookaside buffer.

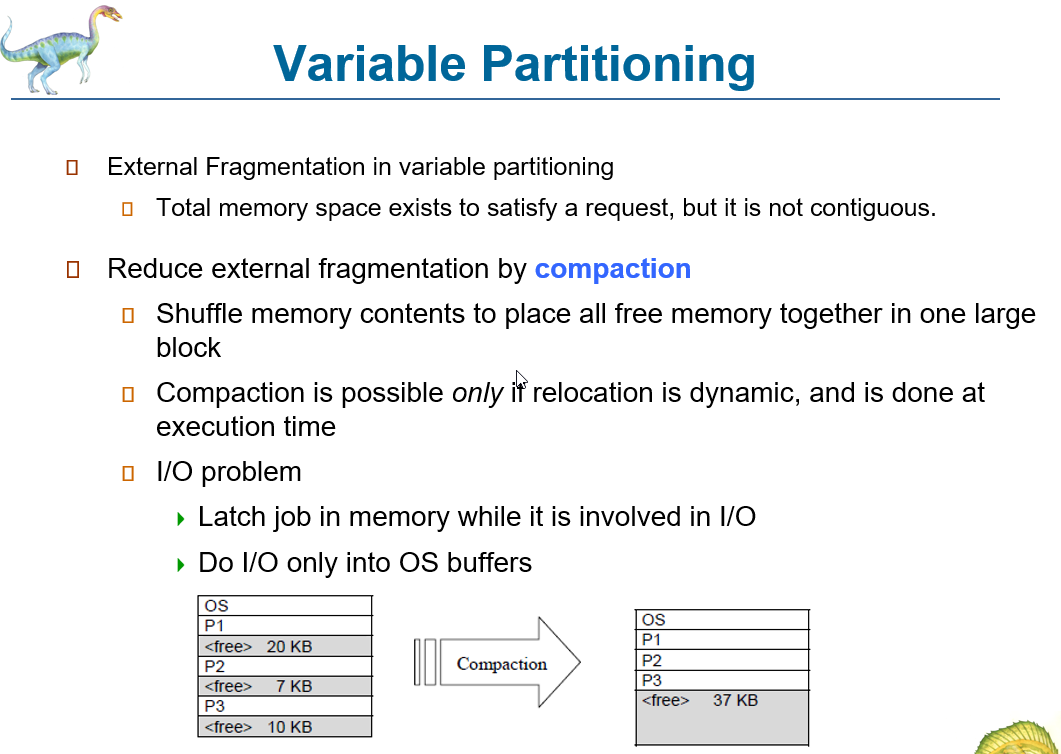
**Ans:**

Translation Lookaside Buffer (TLB) is nothing but a special cache used to keep track of recently used transactions. TLB contains page table entries that have been most recently used. Given a virtual address, the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as index while processing page table. TLB first checks if the page is already in main memory, if not in main memory a page fault is issued then the TLB is updated to include the new page entry.

****

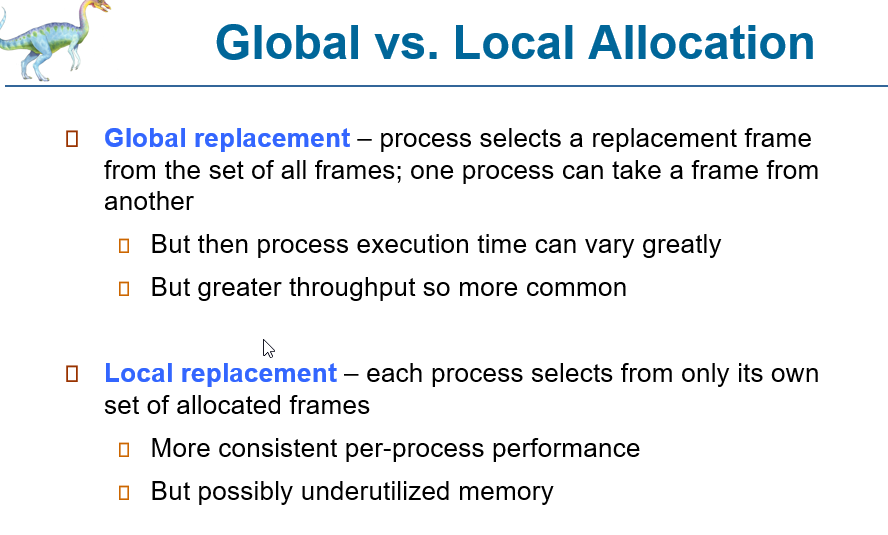
1. **Memory compaction**

**Ans:**

****

1. What are the differences between global and local page replacement techniques?

**Ans:**

****

Explain the Type-1 and Type-2 hypervisors in a virtual machine Operating System.

