# Chapter 9

## LRU-k

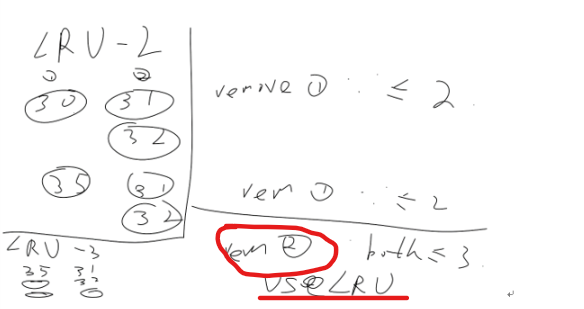
* Example: **2**, 1, 3, **2**, 4, **2**, 4, 1, 5, 6, **2**, …
* Problem
  + Dislodges warm pages if a long sequence of one time page references occur.

EX: Page 2 may get dislodged by the access pattern …, 4, 1, 5, 6,

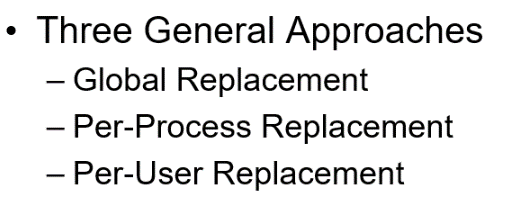
* + LRU does not consider frequency of accesses

LRU-K: Combines recency and frequency

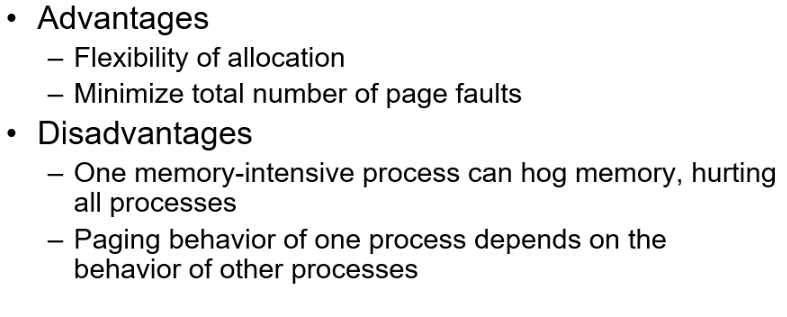
* Idea
  + Keep time of last K references for each page
  + Remove page with oldest Kth reference time
  + If only one page has fewer than K references, remove it
  + If multiple pages have fewer than K references, use LRU
  + LRU-1 is the classic LRU
* Expensive to implement; LRU-2 used in databases



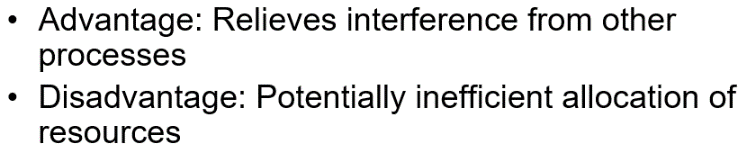
## Allocation



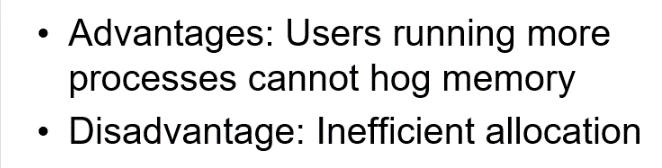
### global



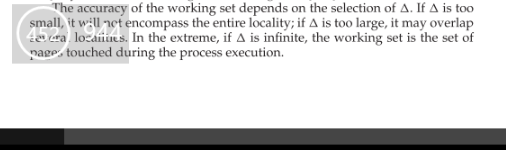
### Per process(?disadvantage)



### Per user



## Working set



## homework

**Consider a system that uses pure demand paging.**

**a. When a process first starts execution, how would you characterize the page-fault rate?**

**b. Once the working set for a process is loaded into memory, how would you characterize the page-fault rate?**

**c. Assume that a process changes its locality and the size of the new working set is too large to be stored in available free memory. Identify some options system designers could choose from to handle this situation.**

a. Because it uses pure demand page, no page has been loaded as a process starts. At this time, page fault number is very high.

b. Once the working set is in memory, the page-fault rate falls because of locality. When the process moves to a new working set, the page-fault rate rises toward a peak once again, returning to a lower rate once the new working set is loaded into memory. The span of time between the start of one peak and the start of the next peak represents the transition from one working set to another.

c. Best way to handle it is to get more physical memory whenever possible. The second way is to reclaim pages more aggressively. The third way is to ignore it.

**LRU and LFU**

LRU requires that the page with smallest count (we keep a counter of the number of references to each page) to be replaced. But the problem is when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed. So when a initially heavily used page appears again after longtime, it will cause a page fault in LRU but not in LFU.

LRU is using past locality to predict future (optimal page replacement looking backward in time). If a certain program keep certain degree of locality over time, LRU should be better than LFU.

**Cause of thrashing**:

The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming as a result. The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing has occurred, and system throughput plunges. **Detect**:

by monitoring the cpu utilization and multiprogramming level.

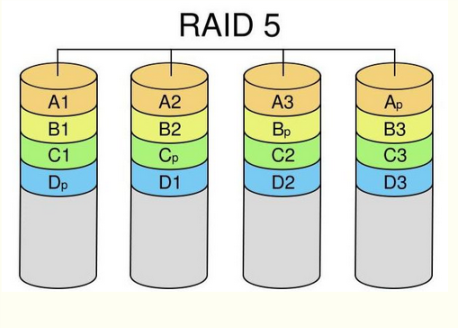
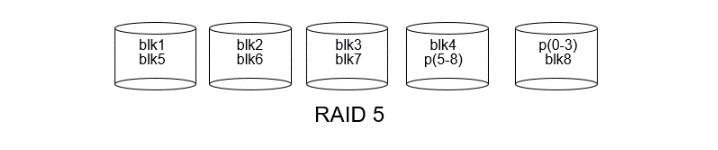
**Eliminate**:

by decreasing the multiprogramming level.

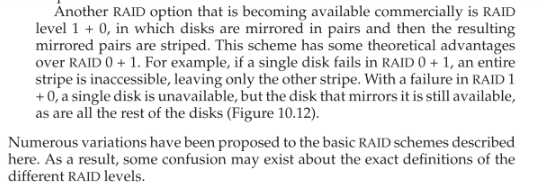
# Chapter 10

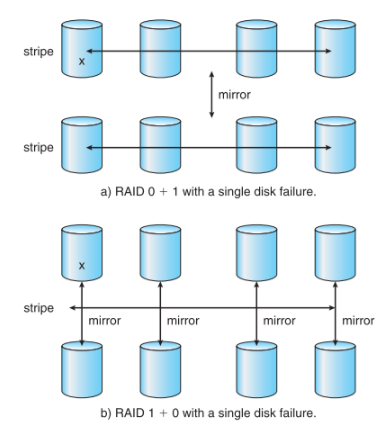
**Mirroring** provides high reliability, but it is expensive. **Striping** provides high data-transfer rates, but it does not improve reliability.

**RAID5**



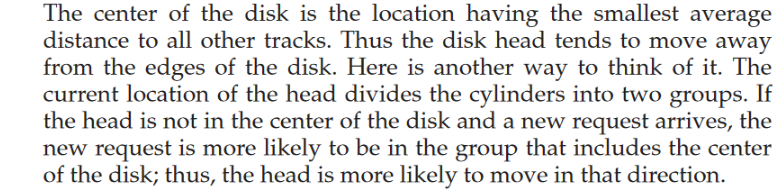
**RAID6**

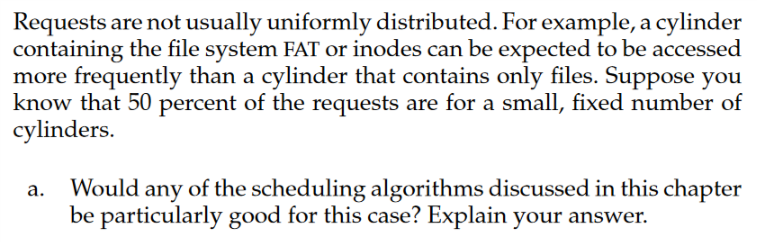


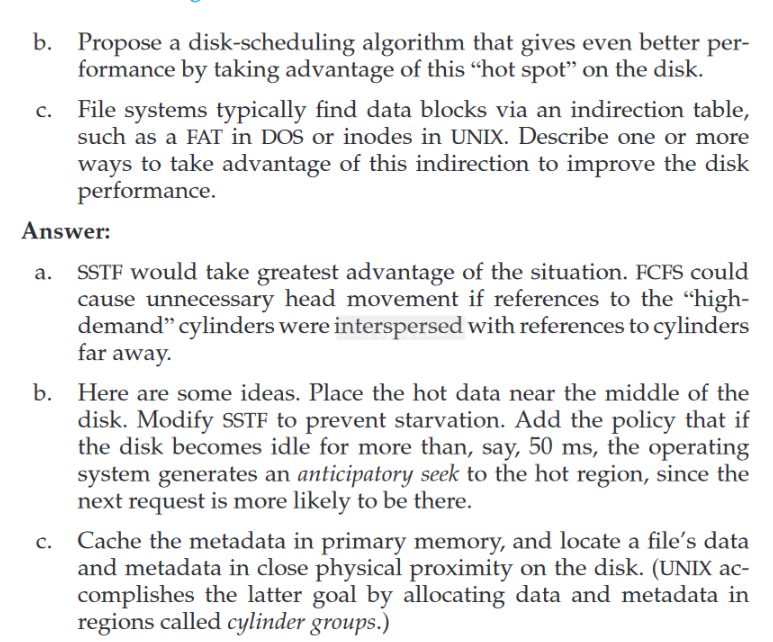


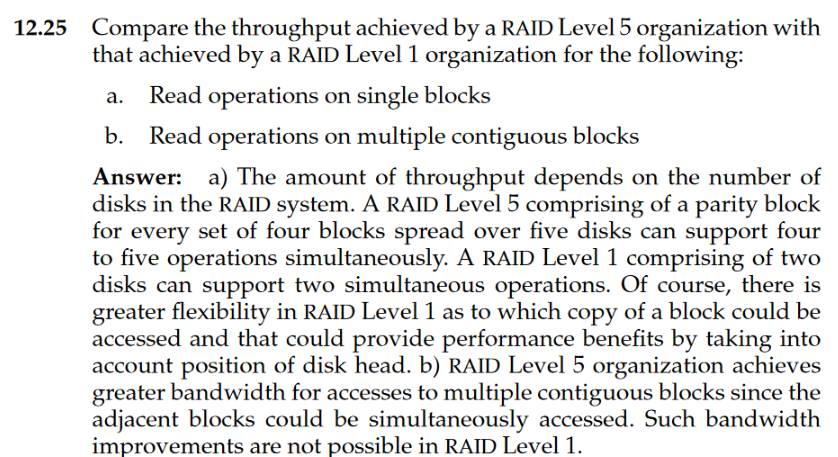
## Homework

**Explain why SSTF scheduling tends to favor middle cylinders over the innermost and outermost cylinders.**



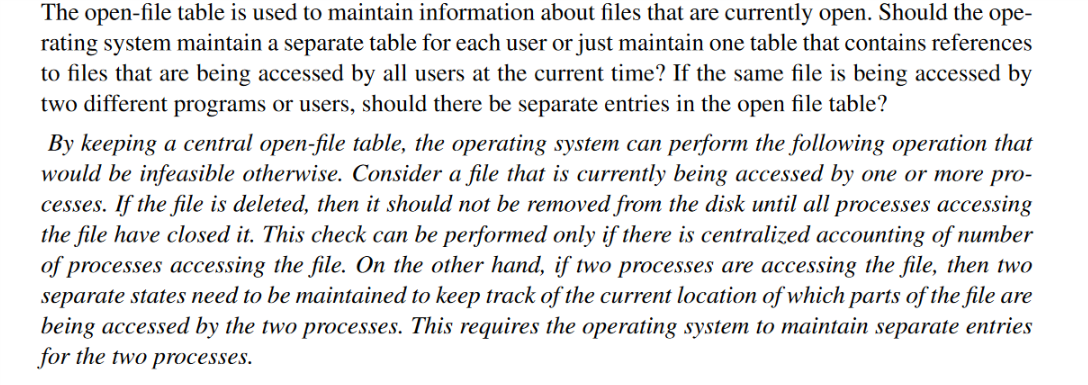


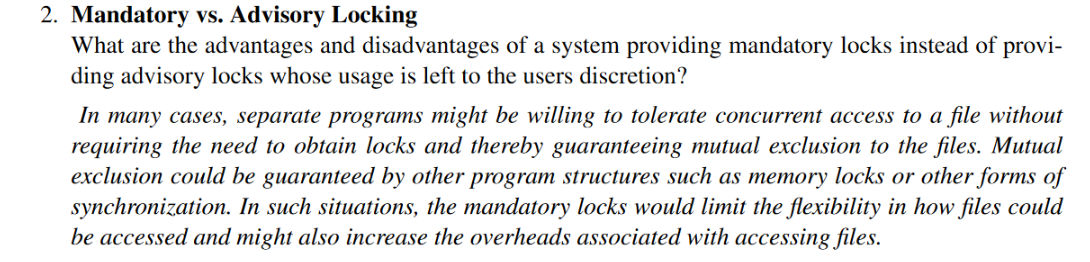


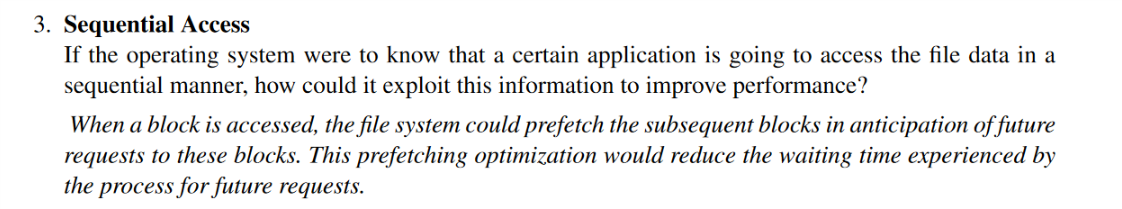


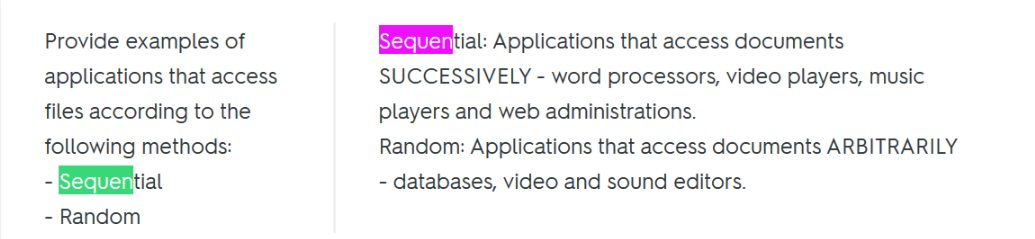
# Chapter11

## Homework





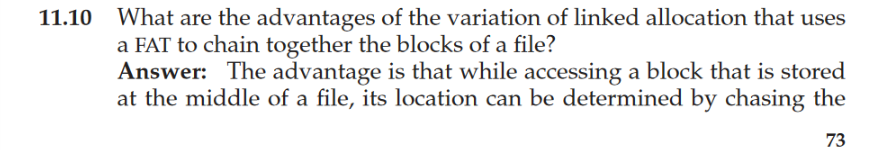
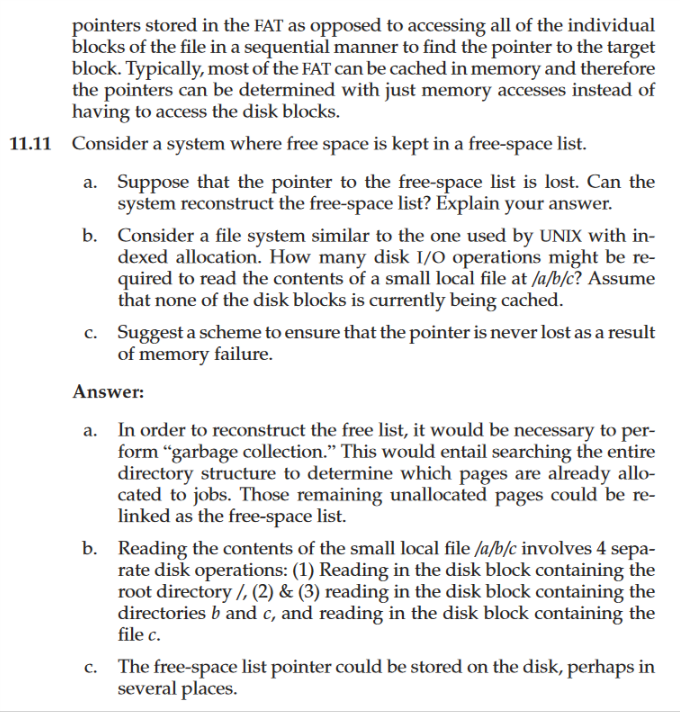


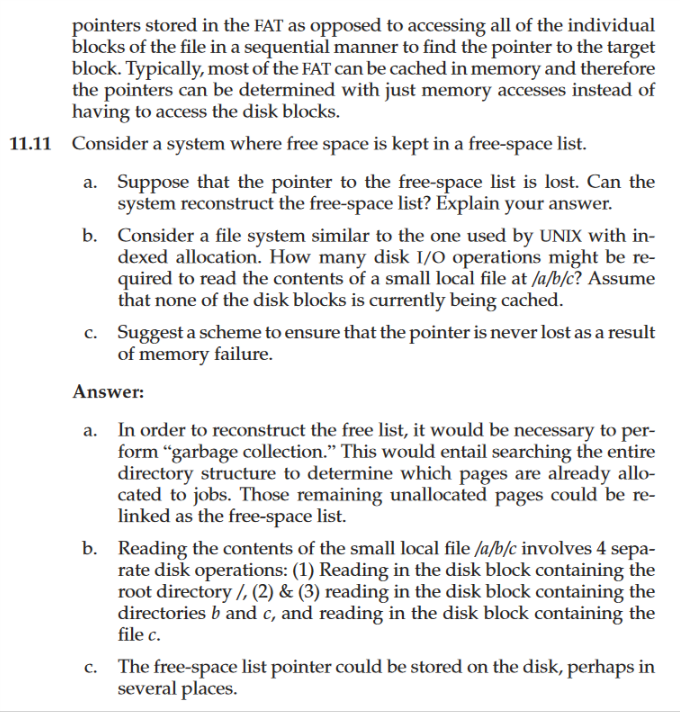


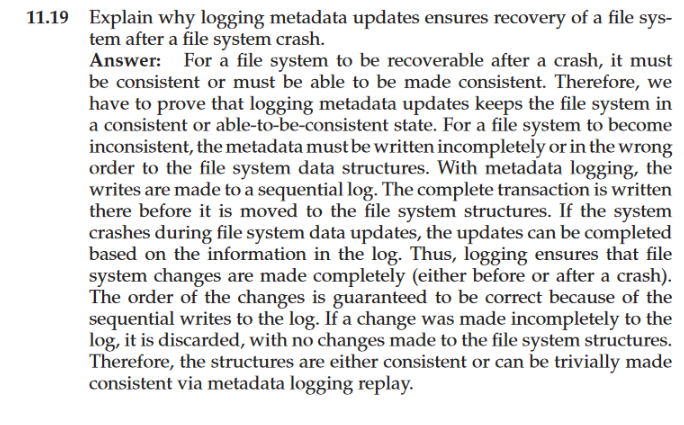
# Chapter12

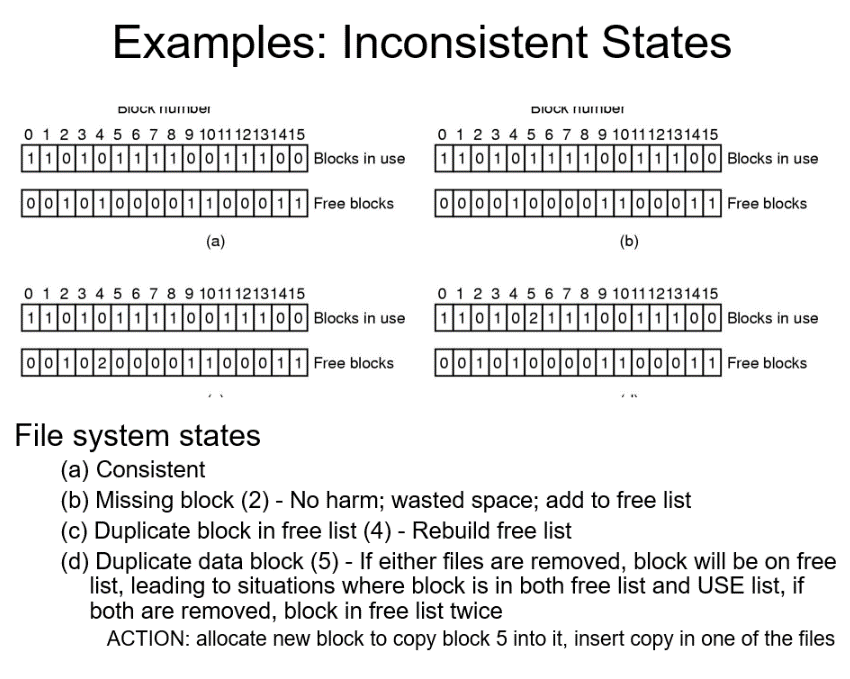
## homework

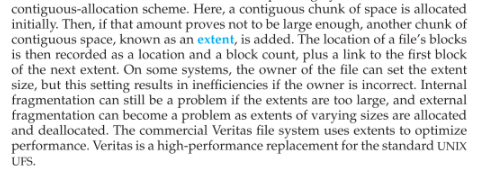
# 









# Chapter 14&15

## Hw

