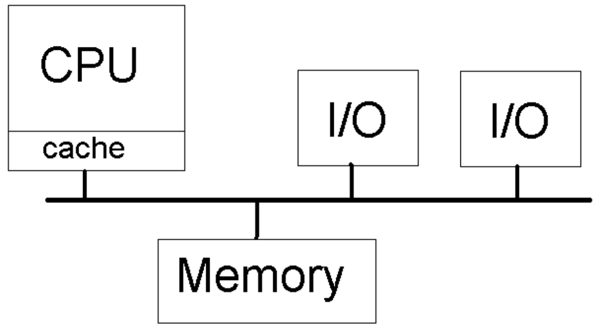
I/O Management

COMP755 Advanced Operating Systems

General Computer Architecture



Goals for I/O

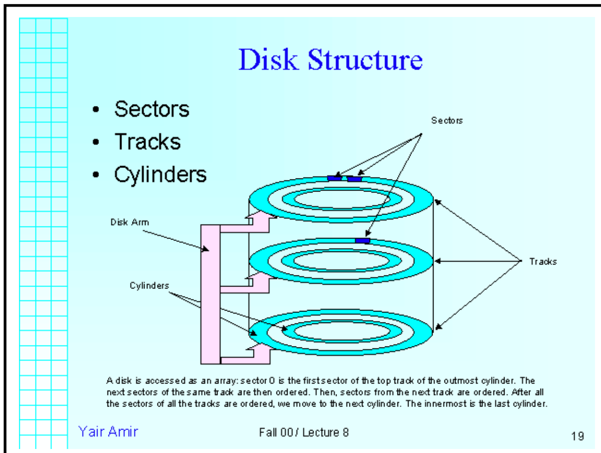
- Users should access all devices in a uniform manner.
- Devices should be named in a uniform manner.
- The OS, without the intervention of the user program, should handle recoverable errors.
- The OS must maintain security of the devices.
- The OS should optimize the performance of the I/O system.

Levels of I/O

- User program
- User level I/O functions
- Device-independent OS software
- Device drivers
- Interrupt handlers

Naming

- In Unix systems, all device names are part of the file address space.
- Hardware devices are “files” in the /dev directory.
- Some Microsoft Windows devices have names such as lpt1 or com2.



Disk Scheduling Policies

- Seek time is the major factor in performance
- There may be a number of I/O requests queued for a device
- Reducing the amount of head motion can significantly improve disk performance

Disk Scheduling Example

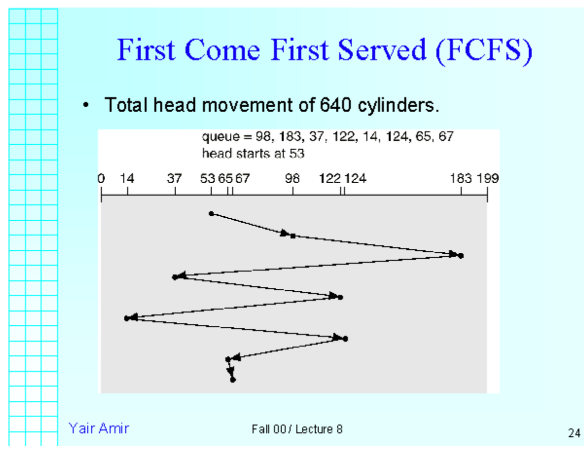
- Assume we have one disk.
- I/O requests to the following tracks have been made in this order

98, 183, 37, 122, 14, 124, 65, 67

- Assume that we start with the disk head at cylinder 53

Disk Scheduling Policies

- First-in, first-out (FIFO)
  - Process request sequentially
  - Fair to all processes
  - Approaches random scheduling in performance if there are many processes

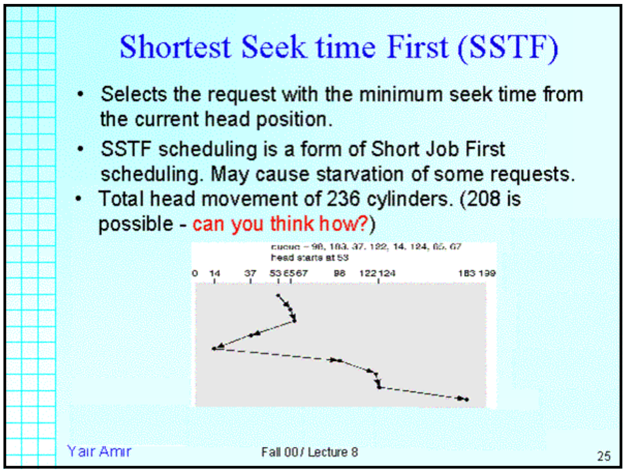


Disk Scheduling Policies

- Last-in, first-out
  - Good for transaction processing systems
    - The device is given to the most recent user so there should be little arm movement
  - Possibility of starvation since a job may never regain the head of the line

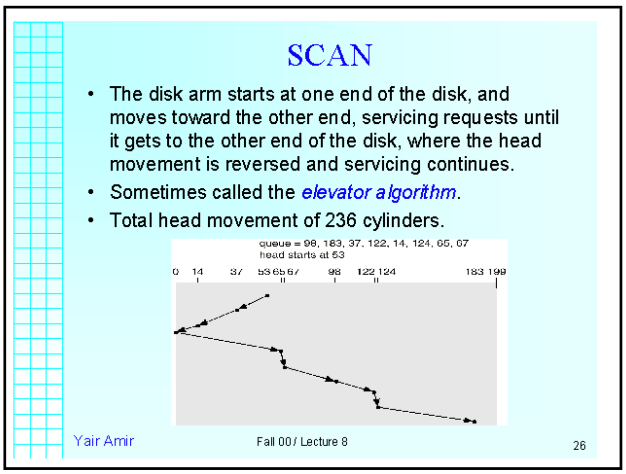
Disk Scheduling Policies

- Shortest Service Time First
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
  - Always choose the minimum Seek time



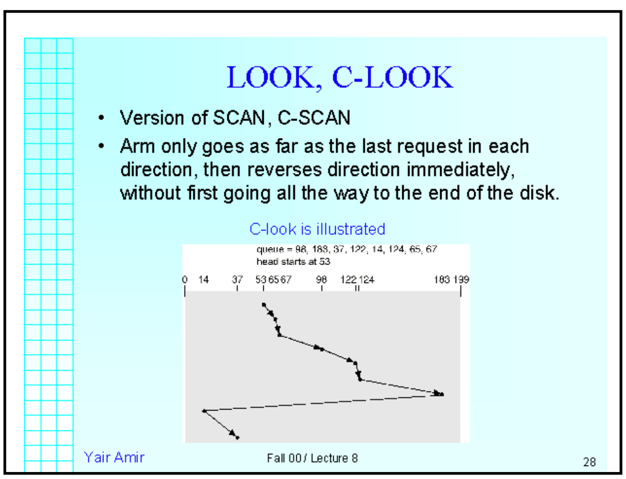
Disk Scheduling Policies

- SCAN
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - Direction is reversed



Disk Scheduling Policies

- C-SCAN
  - Restricts scanning to one direction only
  - When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again



Disk Scheduling Policies

- Priority
  - Goal is not to optimize disk use but to meet other objectives
  - Short batch jobs may have higher priority
  - Provide good interactive response time

Disk Scheduling Policies

- N-step-SCAN
  - Segments the disk request queue into subqueues of length N
  - Subqueues are process one at a time, using SCAN
  - New requests added to other queue when queue is processed
- FSCAN
  - Two queues
  - One queue is empty for new request

Disk Scheduling Algorithms

Table 11.3 Disk Scheduling Algorithms [WIED87]

Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item:		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of *N* records at a time	Service guarantee
FSCAN	N-step-SCAN with *N* = queue size at beginning of SCAN cycle	Load-sensitive

Disk Cache

- RAM is many times faster than the disks. If the OS can keep highly accessed sectors in RAM, it can access them again very quickly.
- A portion of RAM is used to hold disk data.
- RAM used for disk caching competes with application programs for space.

Write Policies

- When a user writes data to a file, the OS can
  - Write Through - Immediately write the data to the disk
  - Write Back - Wait and write the data to the disk during idle periods.

Least Recently Used

- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- Most recently referenced block is on the top of the stack
- When a block is referenced or brought into the cache, it is placed on the top of the stack

Least Frequently Used

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block accessed
- Block with smallest count is selected for replacement
- Some blocks may be referenced many times in a short period of time and then not needed any more

I/O Buffering

- Instead of reading or writing data directly from the user's memory, it is copied to or from an OS buffer
- Reasons for buffering
  - Processes must wait for I/O to complete before proceeding
  - Certain pages must remain in main memory during I/O

I/O Buffering

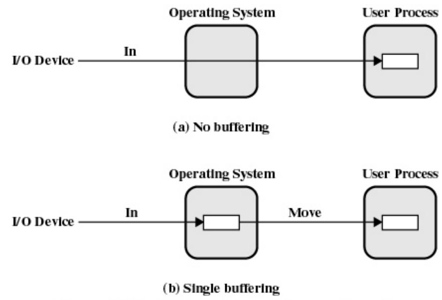


Figure 11.6 I/O Buffering Schemes (input)

Single Buffer

- Operating system assigns a buffer in main memory for an I/O request
- Block-oriented
  - Input transfers made to buffer
  - Block moved to user space when needed
  - Another block is moved into the buffer
    - Read ahead

Double Buffer

- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer
- More than two buffers can be used for circular buffering

I/O Buffering

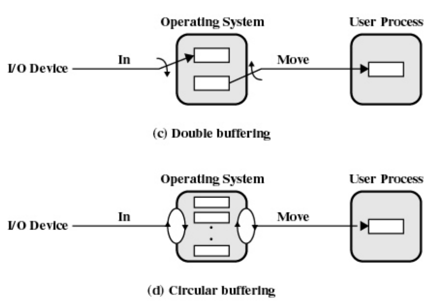


Figure 11.6 I/O Buffering Schemes (input)