Comp 310
Computer Systems and Organization

Lecture #22

Mass-Storage Structures

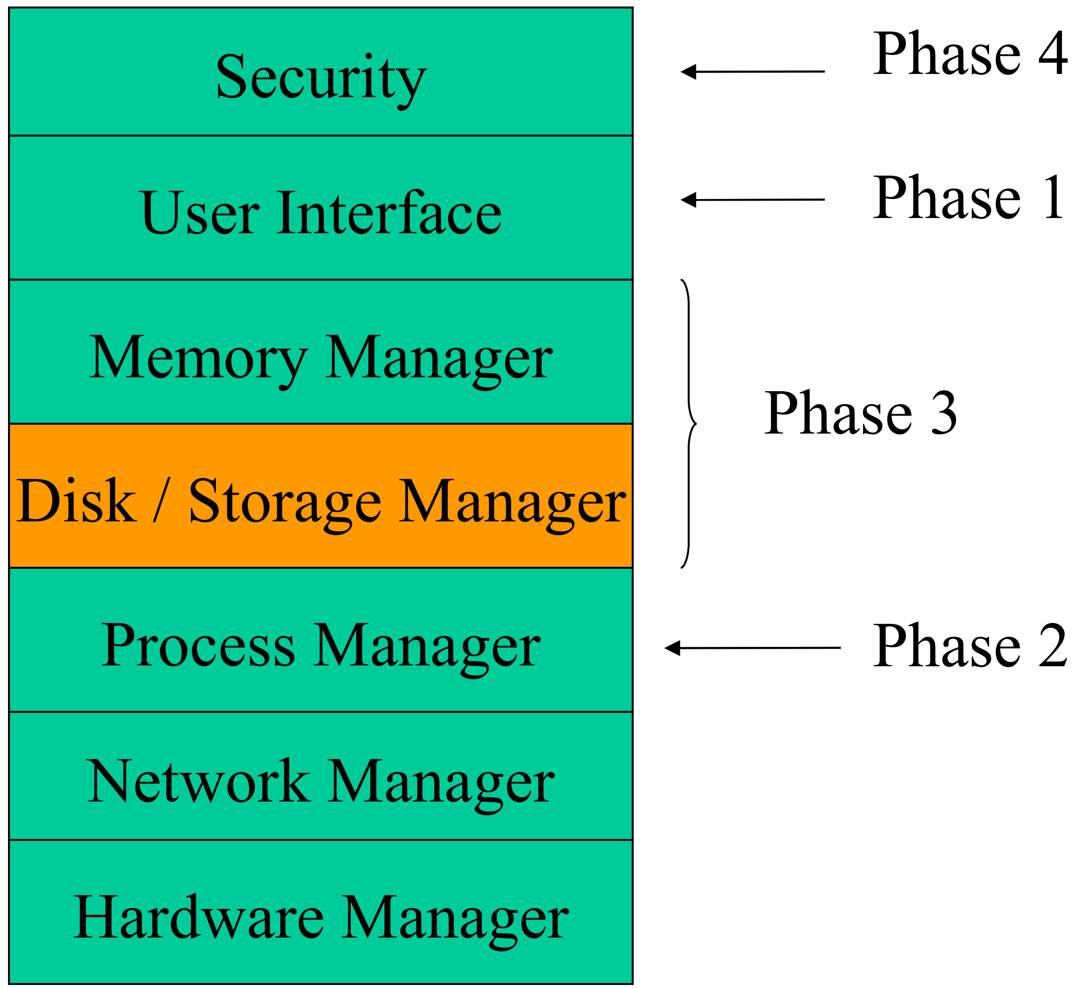
Prof. Joseph Vybihal

Announcements

- **Course Evaluations**

Basic OS Architecture

(Course Table of Contents)

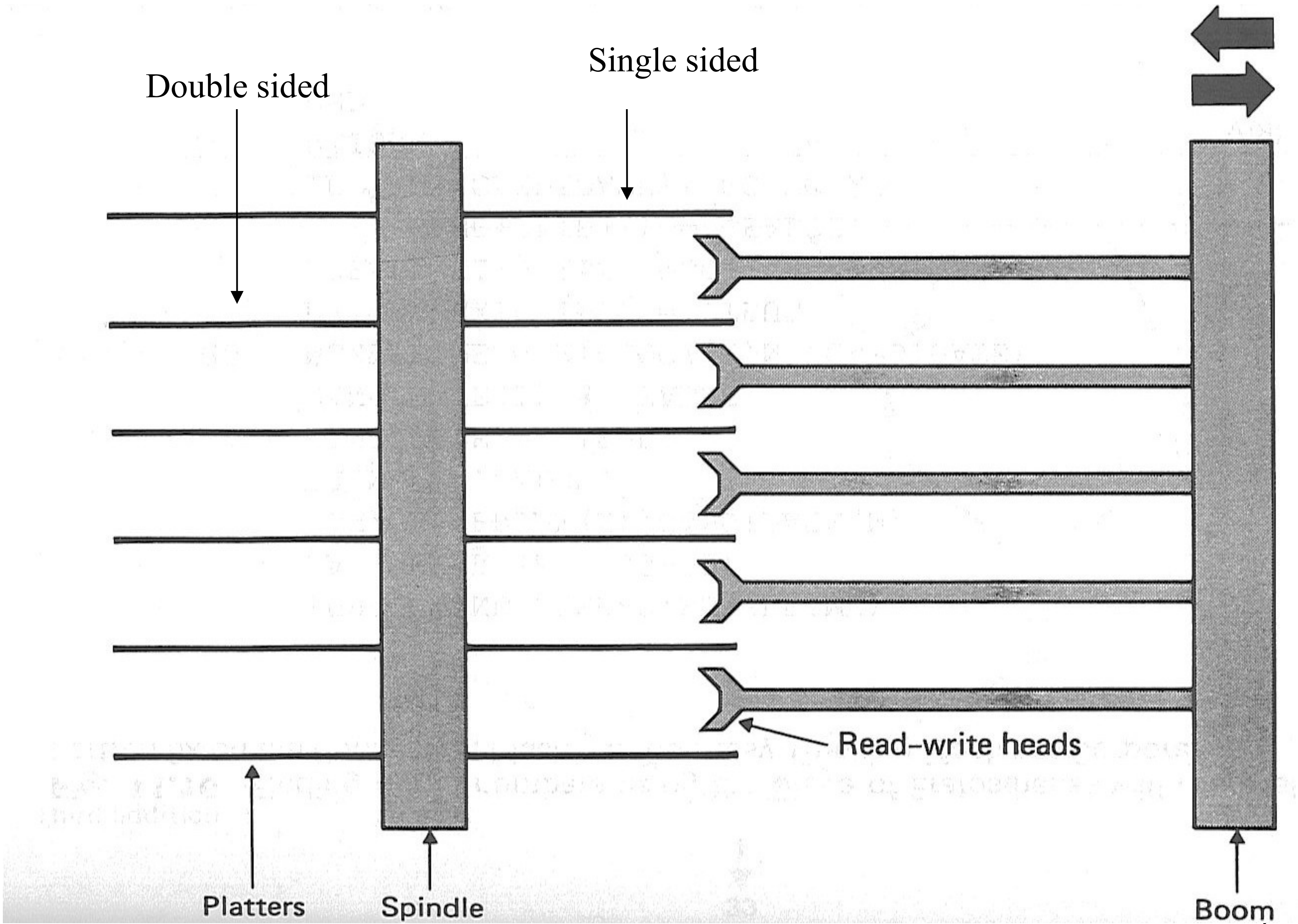


Part 1

Disk Drives & Seek Scheduling

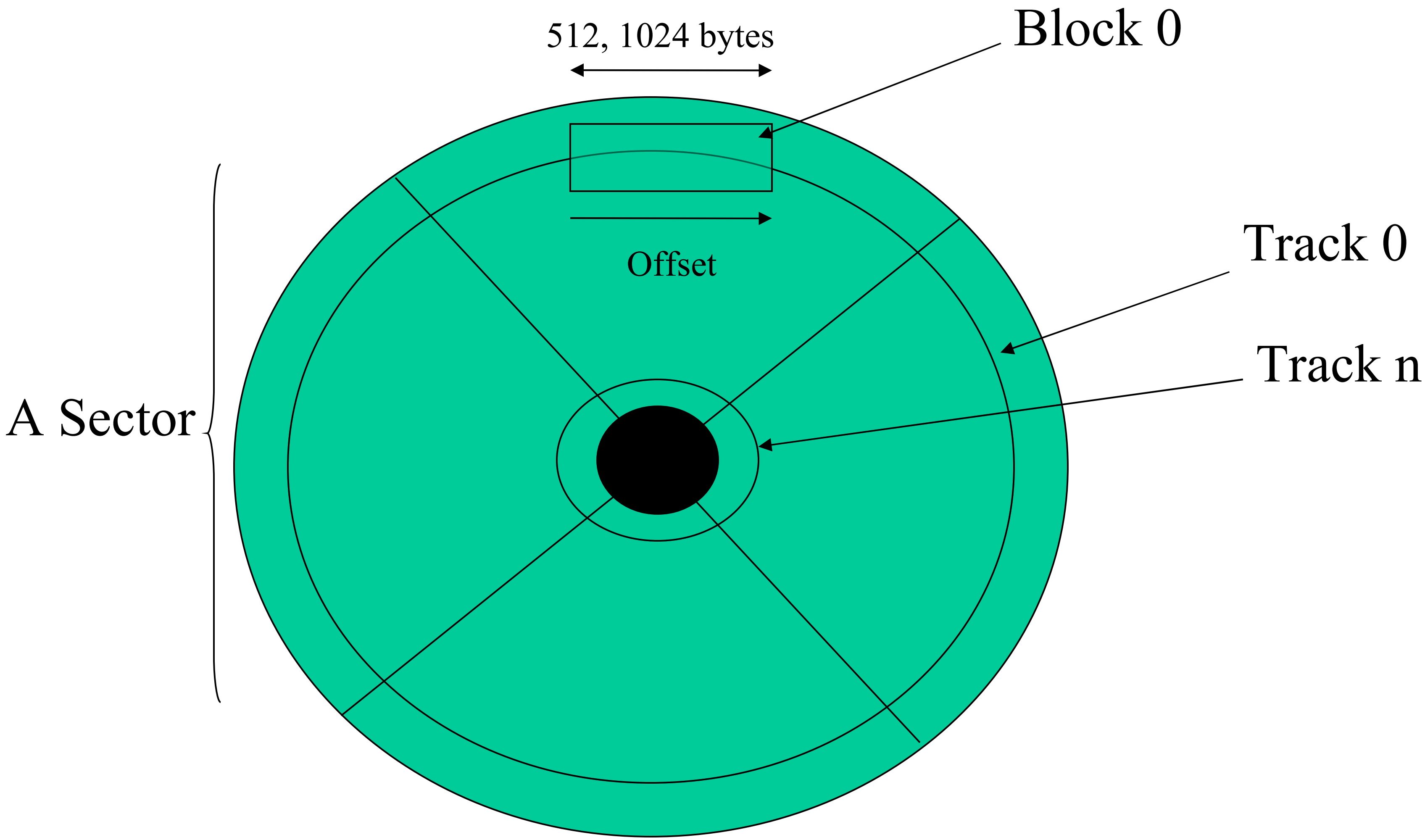
- Addressed as 1-D arrays of logical blocks.
- The logical block is the smallest unit of transfer
- Average block size is 512 bytes

Schematic of Multi-Platter Disk



Formatted Disk

Blocks start, in order, from the outermost track to the innermost.



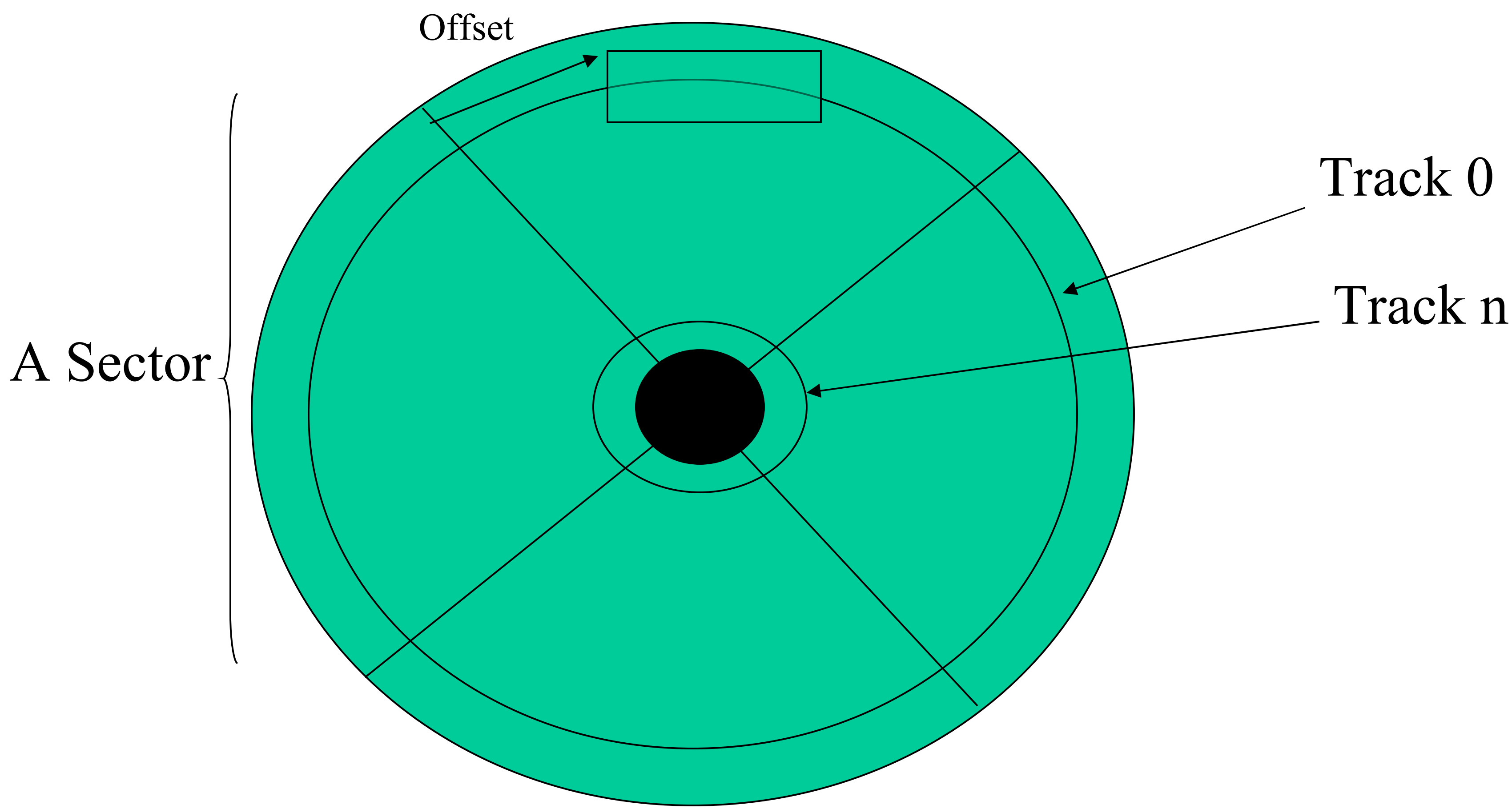
Block Structured

0	1
2	3
4	5
6	7

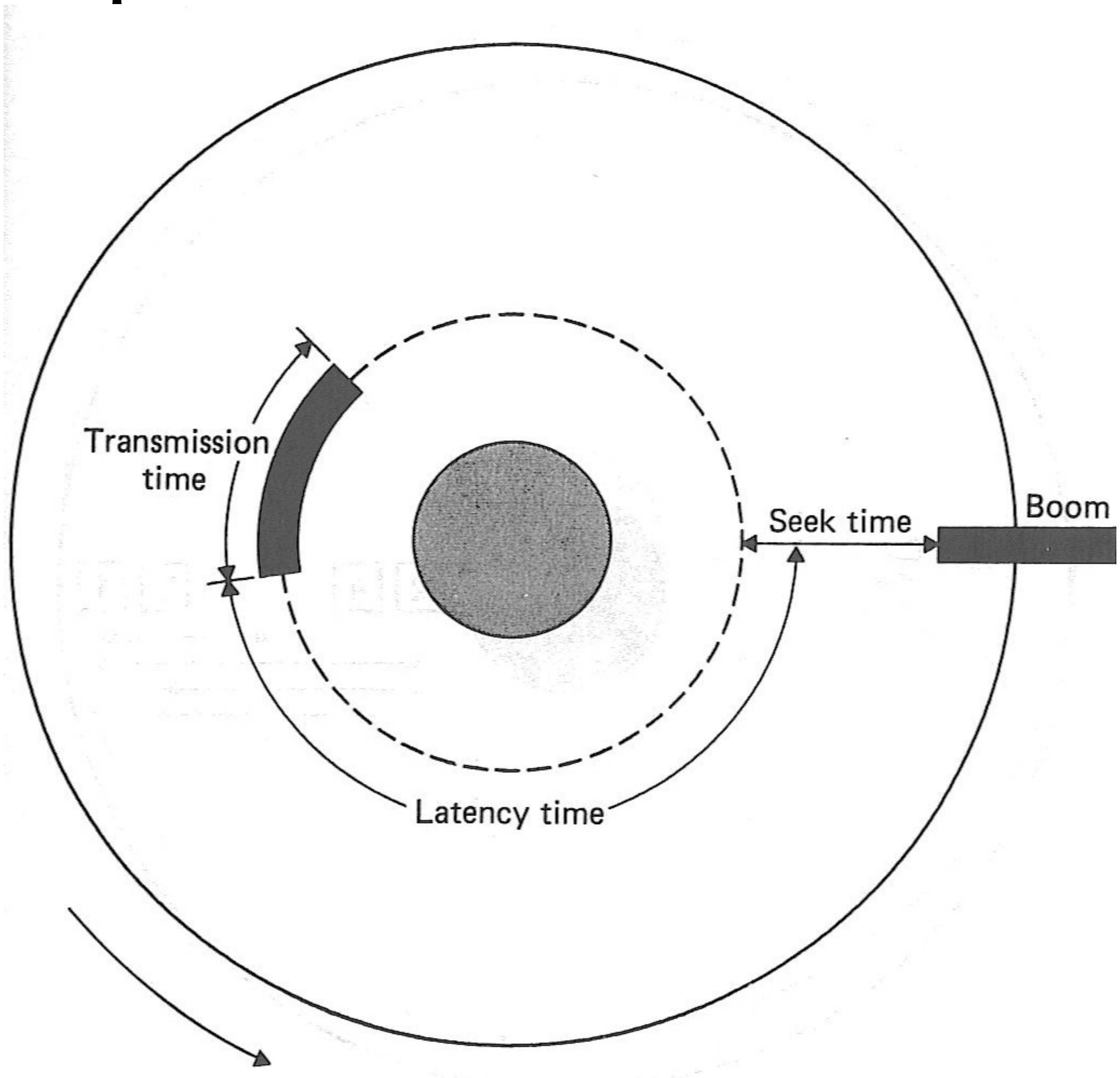
- Entire disk represented as a table of blocks
- Each block of equal size
    - What does that mean for disk allotment
- Easy to index given block number

Formatted Disk: Byte Addressing

Physical address: Track#, Sector#, Offset in bytes from edge of sector.



Components of a Disk Access



Disk Scheduling

- On a single process computer, disk scheduling is not important since only 1 request at any time can be issued.
- In multi-process machines, more than one disk request can be made.
  - Since we know that one disk access is 20 million cycles to complete, optimizing this process is important.
  - We don’t want to waste time seeking when ordering requests could minimize seeks times.

First-come First-served

Not structured

Fig. 12.3 FCFS random seek pattern. The numbers indicate the order in which the requests arrived.

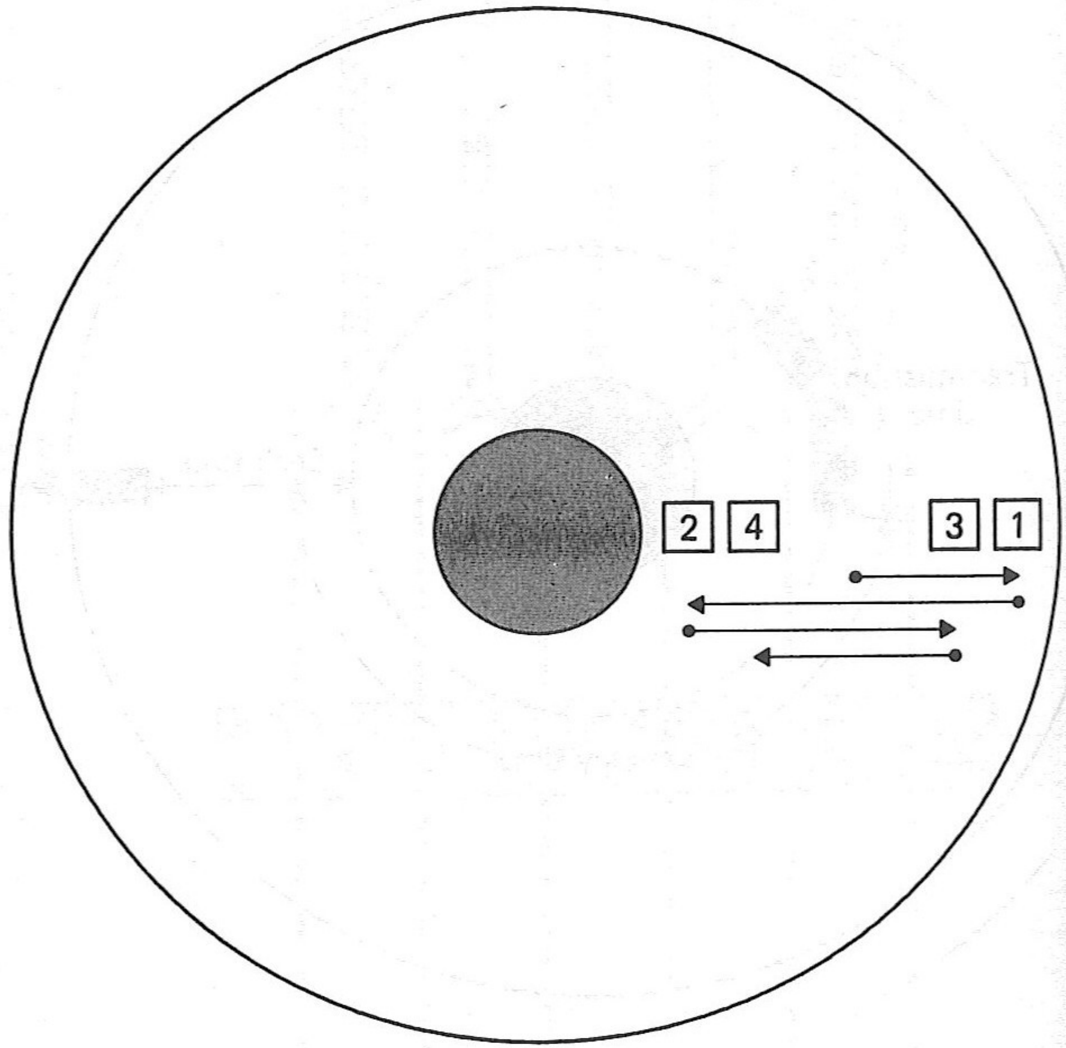
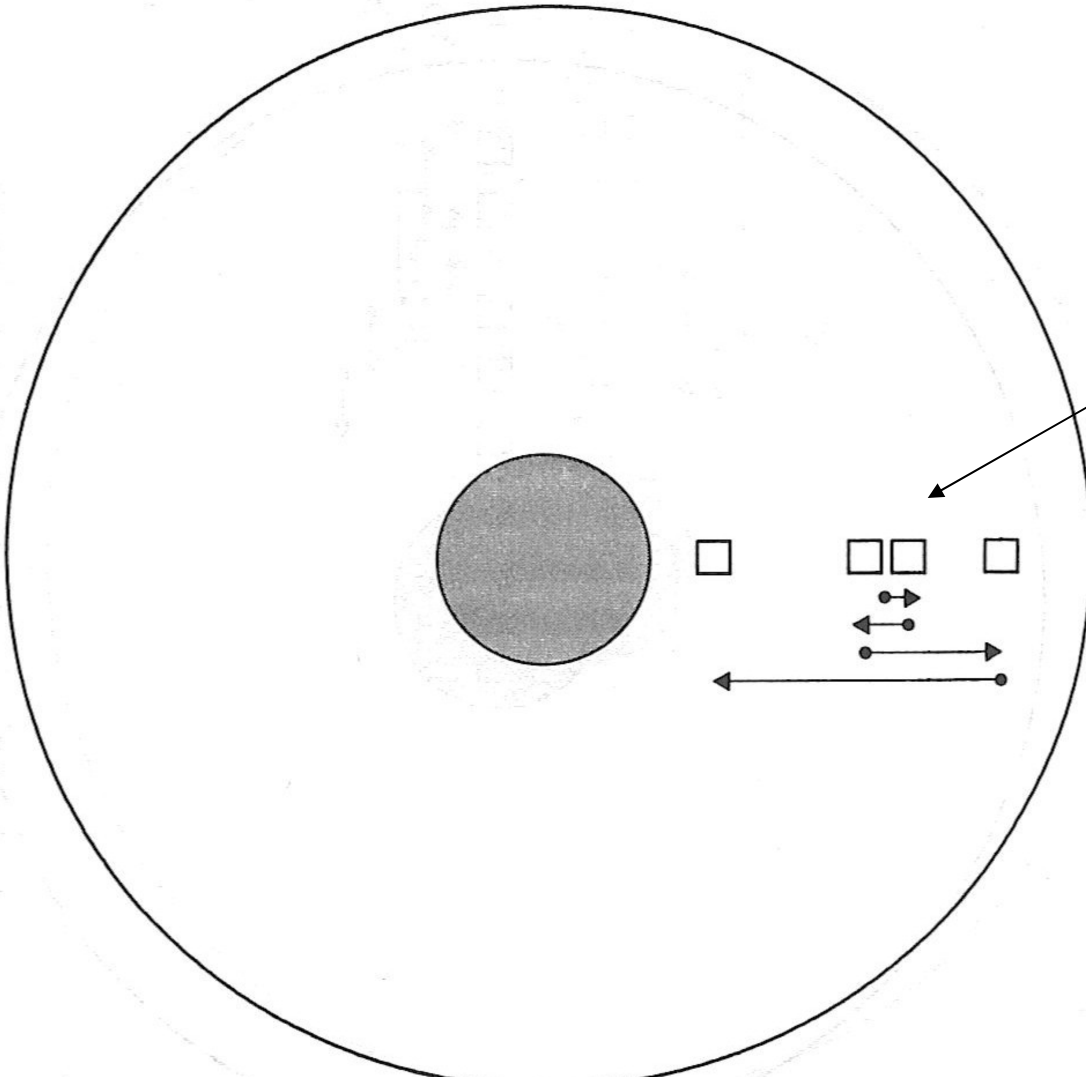


Fig. 12.5 SSTF localized seek pattern.

Shortest Seek Time First

Good, but not optimal



Does not matter in what order the disk requests come, algorithm sorts them.

The Elevator Algorithm

Fig. 12.6 SCAN scheduling with preferred directions.

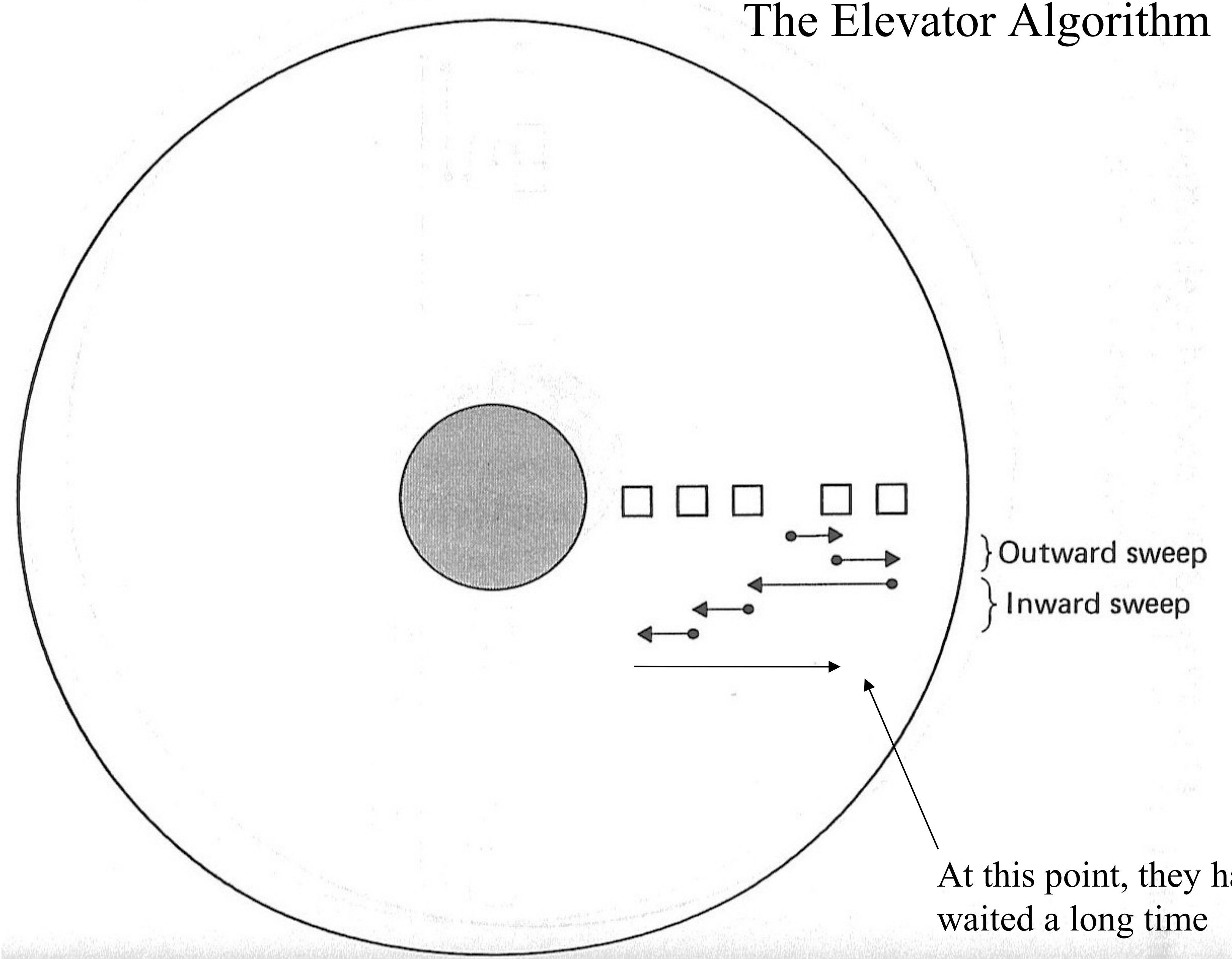
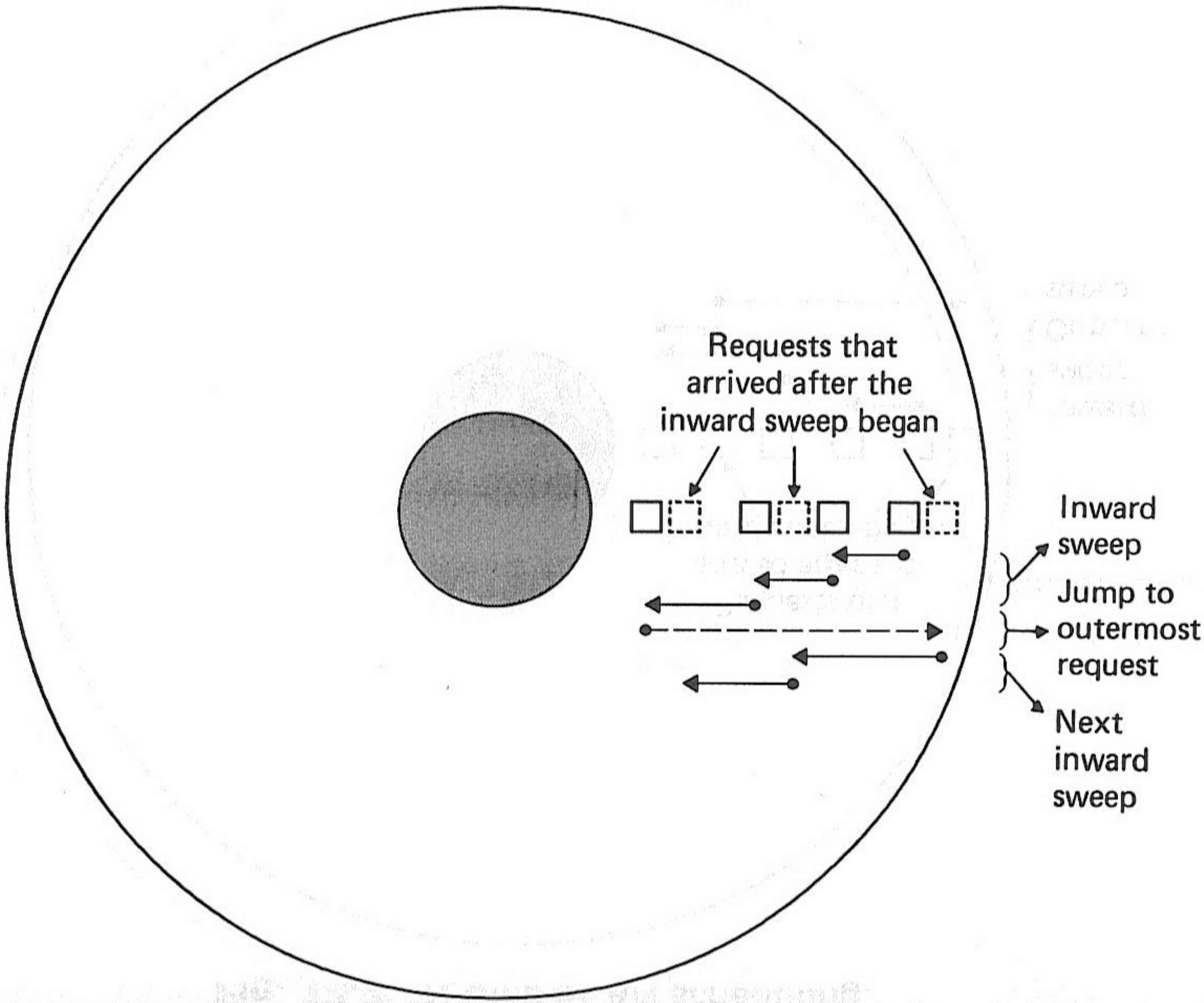


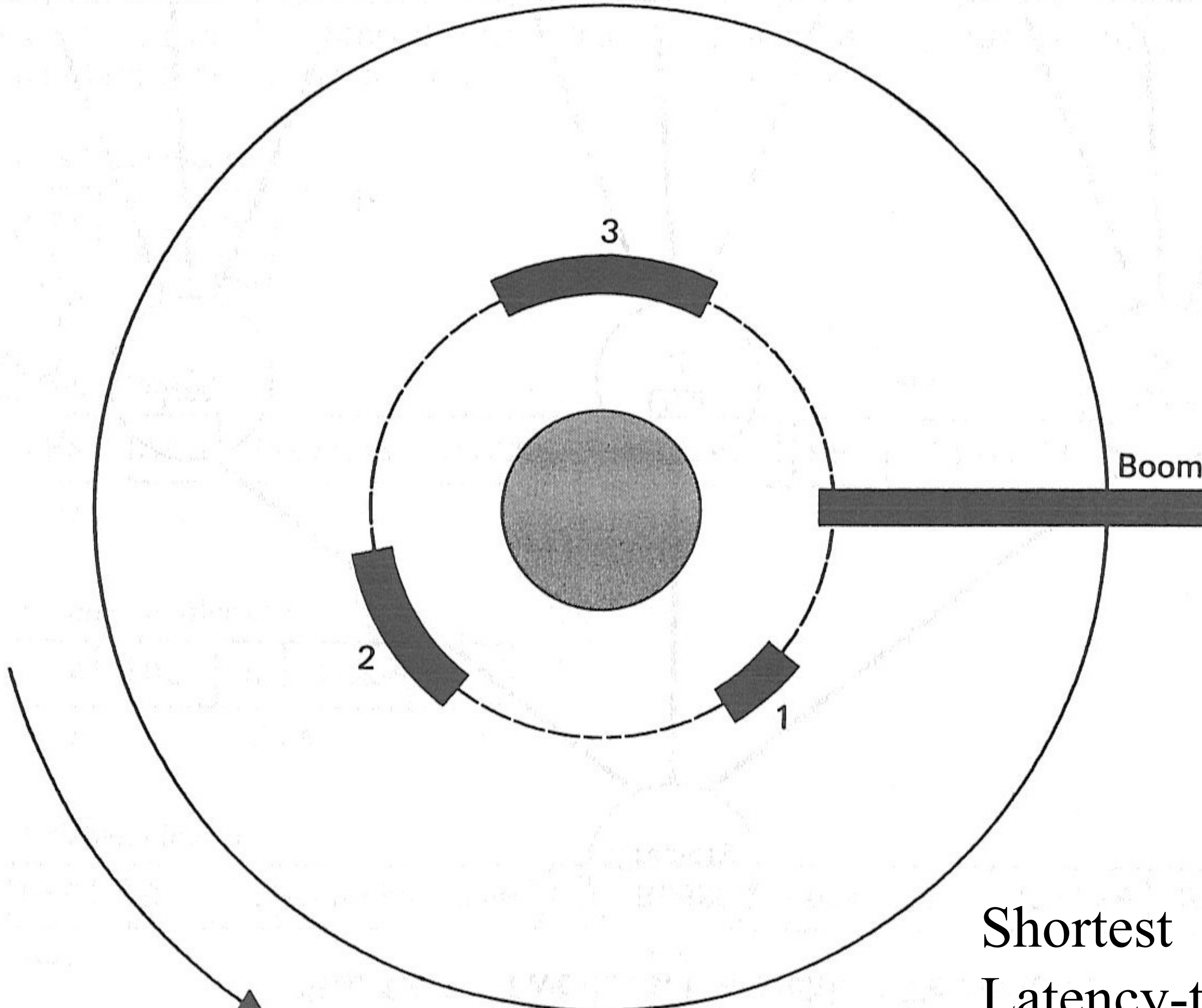
Fig. 12.8 C-SCAN scheduling.



SCAN Variants

- LOOK
  - Unlike SCAN that always goes from track 0 to N and back
  - LOOK goes from min track to max track in queue
- C-LOOK
  - Like C-SCAN but with min/max track in queue

Fig. 12.9 SLTF scheduling. The requests will be serviced in the indicated order regardless of the order in which they arrived.



Shortest Latency-time First

Question

- Using pseudo-coding, how would you implement this?
  - Data structures?
  - Algorithms?

Part 2

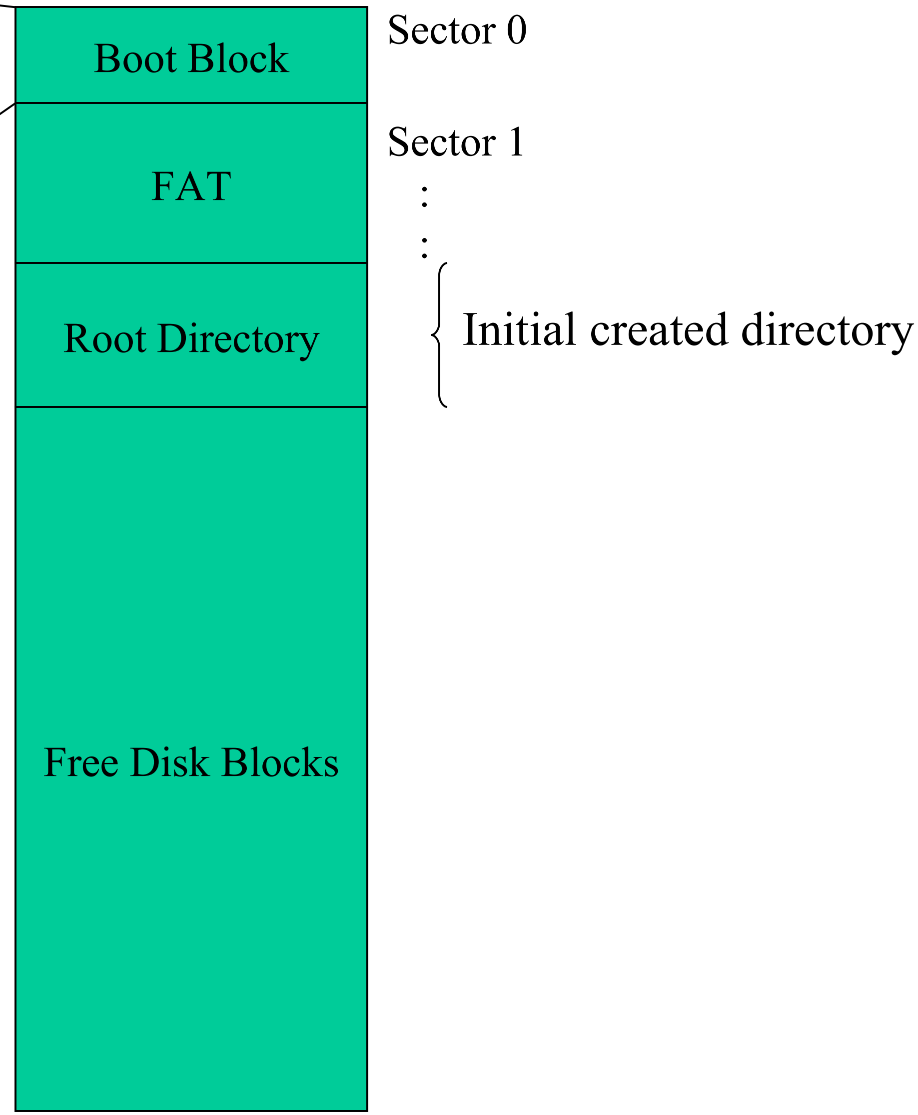
Disk Management

Disk Formatting

- Low-Level Formatting
  - Disk surface structured into tracks and blocks
  - Each block is organized into three sections:
    - Header - sector number
    - Data - the largest and unused area
    - Trailer - Error-correcting code (updated on data change)
    - Block sizes: 256, 512, 1024 bytes
- Partitioning
  - Assigning blocks into a set called a cylinder
  - Each cylinder is viewed as a separate disk
- Logical Formatting
  - File Allocation Table
  - Free Space Table & Bad Block Marker
  - The Operating System (optional)
  - Boot Block (optional)

Formatted Windows Disk Layout

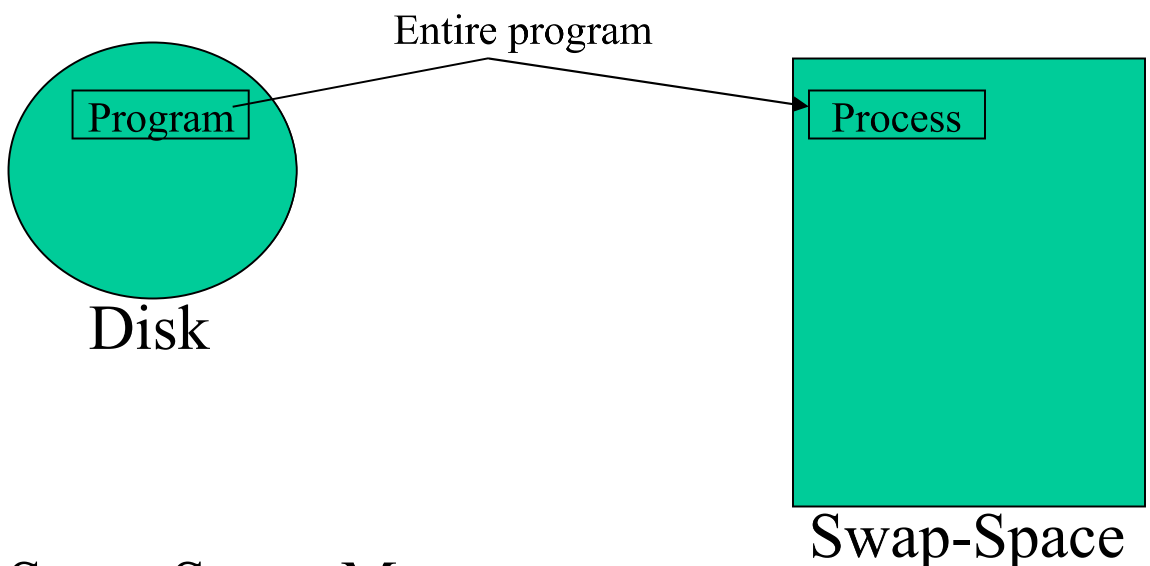
- ROM tells machine to load instruction from block 0
- This is normally the OS, for that cylinder



System Swap-Space

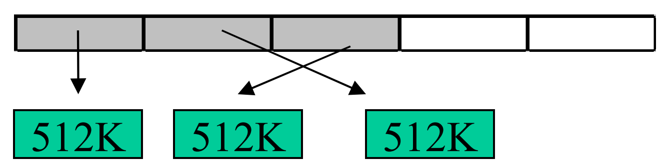
- Needed by virtual memory systems
- Two methods of creation:
  - A large hidden file on a regular cylinder
    - Easy to create (use regular file access commands)
    - Slow access time since in user's complicated FAT
  - A separate partition on the disk
    - Hard to alter size since partitioned to a fixed size
    - Simpler FAT structure, so faster access

Unix BSD Swap-Space



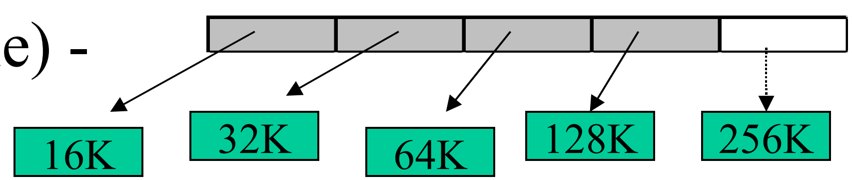
Swap-Space Maps

Text Map (Code, Fixed) -



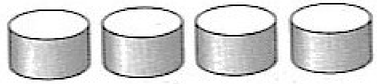
Data Map (Variable) -

As memory is allocated



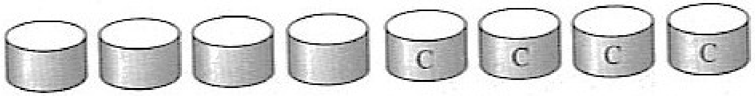
RAID

- Redundant Arrays of Inexpensive Disks
  - High transfer rates
  - High data reliability (to disk crashes)
- Reliability
  - Disk drives fail or get scratched: data lost
  - With many disks can keep duplicate info when storing file. If a disk fails the file can be rebuilt from other disk.
    - Mirroring a disk (save file on more than one disk)
    - Data Stripping (save file across more than one disk)



(a) RAID 0: non-redundant striping

Block level stripping



(b) RAID 1: mirrored disks

Parity - even number of 1 bits (add extra 0 or 1 bit)
(ASCII = 7 bits + 1 parity bit)

(c) RAID 2: memory-style error-correcting codes

File
Restored

(d) RAID 3: bit-interleaved Parity

Use RAID-2 and Sector info to detect & fix bits

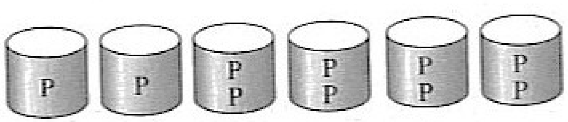
Disk
Restored

(e) RAID 4: block-interleaved parity

Disk's blocks can be recovered
Risk in parity disk loss

(f) RAID 5: block-Interleaved distributed parity

Disk's block restorable as well as parity bits



(g) RAID 6: P + Q redundancy

Like RAID-5 but handles multi-disk failures

Question

- Using pseudo-code, how would you format a hard disk?
  - Quick format?
  - Complete format?
  - Complete erase format?
- How could we code faster disk access:
  - For shared files?
  - For different files?
  - Combined features?

Questions

- How could we implement “community” directories...

Part 3

At Home

Things to try out

1. Run a single process that does a lot of disk access and time it to completion.
   - Repeat this with two processes that perform a lot of disk accesses
   - How much slower was the first process?
2. Web Resources:
   - http://www.lvr.com/mass_storage.htm
   - http://support.pa.msu.edu/Help/FAQs/Linux/harddisks.html
   - http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/t0004971.htm
   - http://support.microsoft.com/kb/q140372/