

CS162
Operating Systems and
Systems Programming
Lecture 13

Disk/SSDs,
File Systems (Part 1)

October 16, 2013
Anthony D. Joseph and John Canny
<http://inst.eecs.berkeley.edu/~cs162>

Quiz 12.1: I/O

- Q1: True _ False _ With an asynchronous interface, the writer may need to block until the data is written
- Q2: True _ False _ Interrupts are more efficient than polling for handling very frequent requests
- Q3: True _ False _ Segmentation fault is an example of synchronous exception (trap)
- Q4: True _ False _ DMA is more efficient than programmed I/O for transferring large volumes of data
- Q5: In a I/O subsystem the queuing time for a request is 10ms and the request's service time is 40ms. Then the total response time of the request is ____ ms

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.2

Quiz 12.1: I/O

- Q1: True _ False **x** With an asynchronous interface, the writer may need to block until the data is written
- Q2: True _ False **x** Interrupts are more efficient than polling for handling very frequent requests
- Q3: True **x** False _ Segmentation fault is an example of synchronous exception (trap)
- Q4: True **x** False _ DMA is more efficient than programmed I/O for transferring large volumes of data
- Q5: In a I/O subsystem the queuing time for a request is 10ms and the request's service time is 40ms. Then the total response time of the request is **50** ms

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.3

Quiz 13.1: Synchronization

- Q1: True _ False _ During a critical section, a thread can be preempted by the CPU dispatcher
- Q2: True _ False _ If we use interrupts to implement locks we need to enable interrupts before going to sleep (in the lock() primitive)
- Q3: True _ False _ The order of sem.P() and sem.V() in a program is commutative
- Q4: True _ False _ With Mesa monitors, the program needs to check again the condition (on which it went to sleep) after waking up
- Q5: True _ False _ In a database (think of the Readers/Writers problem), a user can read while another one writes

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.4

Quiz 13.1: Synchronization

- Q1: True ☒ False ☐ During a critical section, a thread can be preempted by the CPU dispatcher
- Q2: True ☐ False ☒ If we use interrupts to implement locks we need to enable interrupts before going to sleep (in the lock() primitive)
- Q3: True ☒ False ☐ The order of sem.P() and sem.V() in a program is commutative
- Q4: True ☒ False ☐ With Mesa monitors, the program needs to check again the condition (on which it went to sleep) after waking up
- Q5: True ☐ False ☒ In a database (think of the Readers/Writers problem), a user can read while another one writes

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.5

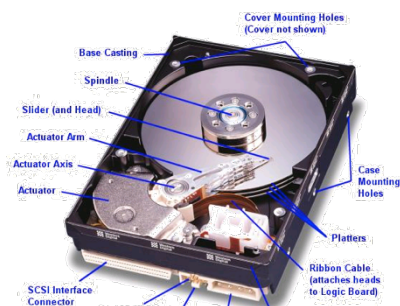
Goals for Today

- Disks and SSDs
- Important Storage Policies and Patterns
- File Systems Structures

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne. Many slides generated from my lecture notes by Kubiawicz.

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.6

Hard Disk Drives (HDDs)



Western Digital Drive
<http://www.storagereview.com/guide/>

IBM Personal Computer/AT (1986)
 30 MB hard disk - \$500
 30-40ms seek time
 0.7-1 MB/s (est.)



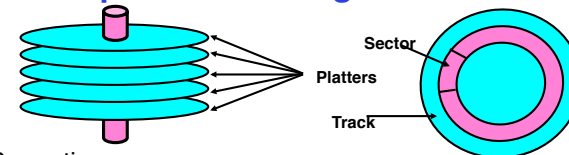
Read/Write Head
Side View



IBM/Hitachi Microdrive

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.7

Properties of a Magnetic Hard Disk

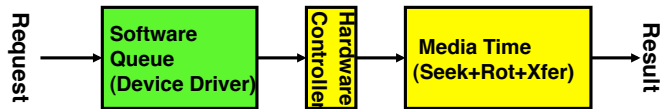


- Properties
 - Independently addressable element: **sector**
 - » OS always transfers groups of sectors together – “blocks”
 - A disk can access directly any given block either sequentially or randomly.
- Typical numbers (depending on the disk size):
 - 500 to more than 20,000 tracks per surface
 - 32 to 800 sectors per track
- Zoned bit recording
 - Constant bit density: more bits (sectors) on outer tracks
 - Apple][gs/old Macs: speed varies with track location

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.8

Magnetic Disk Characteristic

- Cylinder: all the tracks under the head at a given point on all surfaces
- Read/write: three-stage process:
 - **Seek time**: position the head/arm over the proper track (into proper cylinder)
 - **Rotational latency**: wait for the desired sector to rotate under the read/write head
 - **Transfer time**: transfer a block of bits (sector) under the read-write head
- **Disk Latency = Queuing Time + Controller time + Seek Time + Rotation Time + Xfer Time**



- **Highest Bandwidth**:
 - Transfer large group of blocks sequentially from one track

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.9

Typical Numbers of a Magnetic Disk

Parameter	Info / Range
Average seek time	Typically 5-10 milliseconds. Depending on reference locality, actual cost may be 25-33% of this number.
Average rotational latency	Most laptop/desktop disks rotate at 3600-7200 RPM (16-8 ms/rotation). Server disks up to 15,000 RPM. Average latency is halfway around disk yielding corresponding times of 8-4 milliseconds
Controller time	Depends on controller hardware
Transfer time	Typically 50 to 100 MB/s. Depends on: <ul style="list-style-type: none"> • Transfer size (usually a sector): 512B – 1KB per sector • Rotation speed: 3600 RPM to 15000 RPM • Recording density: bits per inch on a track • Diameter: ranges from 1 in to 5.25 in
Cost	Drops by a factor of two every 1.5 years (or even faster). \$0.05/GB in 2012

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.10

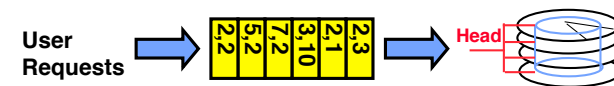
Disk Performance Examples

- Assumptions:
 - Ignoring queuing and controller times for now
 - Avg seek time of 5ms,
 - 7200RPM \Rightarrow Time for one rotation: $60000\text{ms}/7200 \approx 8\text{ms}$
 - Transfer rate of 4MByte/s, sector size of 1 KByte
- Read sector from random place on disk:
 - Seek (5ms) + Rot. Delay (4ms) + Transfer (0.25ms)
 - Approx 10ms to fetch/put data: **100 KByte/sec**
- Read sector from random place in same cylinder:
 - Rot. Delay (4ms) + Transfer (0.25ms)
 - Approx 5ms to fetch/put data: **200 KByte/sec**
- Read next sector on same track:
 - Transfer (0.25ms): **4 MByte/sec**
- **Key to using disk effectively (especially for file systems) is to minimize seek and rotational delays**

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.11

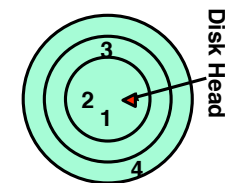
Disk Scheduling

- Disk can do only one request at a time; What order do you choose to do queued requests?
 - Request denoted by (track, sector)



- Scheduling algorithms:
 - First In First Out (FIFO)
 - Shortest Seek Time First
 - SCAN
 - C-SCAN

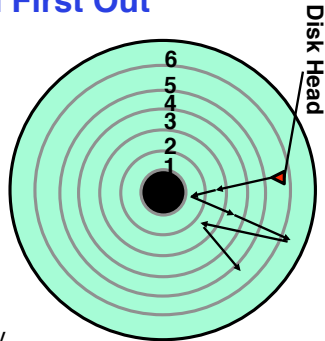
- In our examples we will ignore the sector
 - Consider only track #



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.12

FIFO: First In First Out

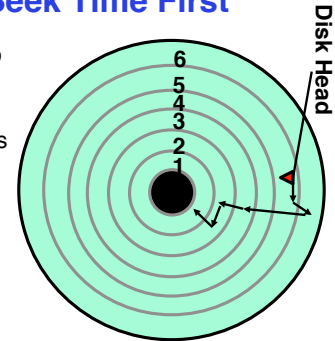
- Schedule request in the order they arrive in the queue
- Example:
 - Request queue: 2, 1, 3, 6, 2, 5
 - Scheduling order: 2, 1, 3, 6, 2, 5
- Pros: Fair among requesters
- Cons: Order of arrival may be to random spots on the disk \Rightarrow Very long seeks



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.13

SSTF: Shortest Seek Time First

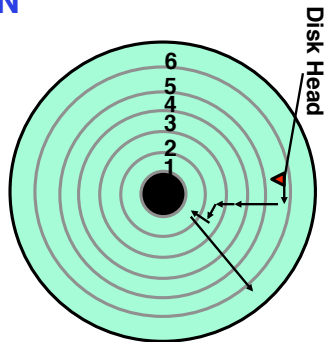
- Pick the request that's closest to the head on the disk
 - Although called SSTF, include rotational delay in calculation, as rotation can be as long as seek
- Example:
 - Request queue: 2, 1, 3, 6, 2, 5
 - Scheduling order: 5, 6, 3, 2, 2, 1
- Pros: reduce seeks
- Cons: may lead to starvation



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.14

SCAN

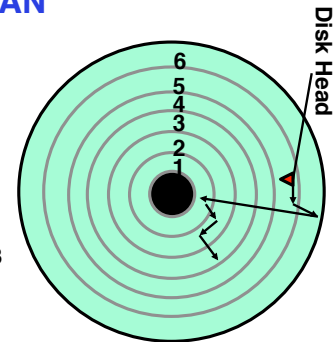
- Implements an Elevator Algorithm: take the closest request in the direction of travel
- Example:
 - Request queue: 2, 1, 3, 6, 2, 5
 - Head is moving towards center
 - Scheduling order: 5, 3, 2, 2, 1, 6
- Pros:
 - No starvation
 - Low seek
- Cons: favor middle tracks



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.15

C-SCAN

- Like SCAN but only serves request in only one direction
- Example:
 - Request queue: 2, 1, 3, 6, 2, 5
 - Head only serves request on its way from center towards edge
 - Scheduling order: 5, 6, 1, 2, 2, 3
- Pros:
 - Fairer than SCAN
- Cons: longer seeks on the way back



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.16

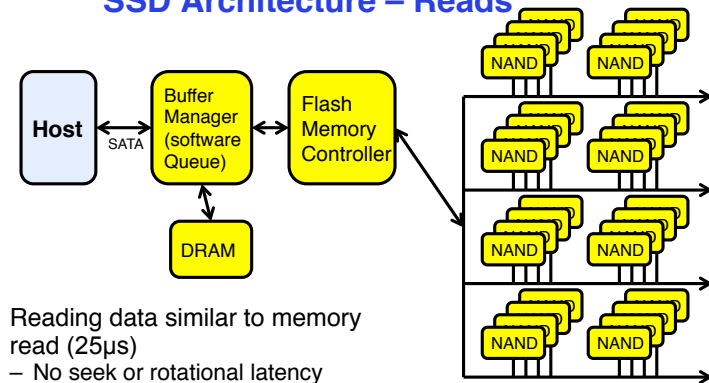
Solid State Disks (SSDs)



- 1995 – Replace rotating magnetic media with non-volatile memory (battery backed DRAM)
- 2009 – Use NAND Multi-Level Cell (2-bit/cell) flash memory
 - Sector (4 KB page) addressable, but stores 4-64 “pages” per memory block
- No moving parts (no rotate/seek motors)
 - Eliminates seek and rotational delay (0.1-0.2ms access time)
 - Very low power and lightweight

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.17

SSD Architecture – Reads



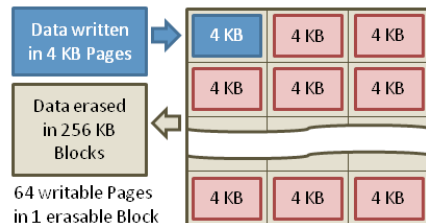
Reading data similar to memory read (25μs)

- No seek or rotational latency
- Transfer time: transfer a 4KB page
 - » Limited by controller and disk interface (SATA: 300-600MB/s)
- Latency = Queuing Time + Controller time + Xfer Time
- Highest Bandwidth: Sequential OR Random reads

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.18

SSD Architecture – Writes (I)

- Writing data is complex! (~200μs – 1.7ms)
 - Can only write empty pages in a block
 - Erasing a block takes ~1.5ms
 - Controller maintains pool of empty blocks by coalescing used pages (read, erase, write), also reserves some % of capacity



Typical NAND Flash Pages and Blocks

https://en.wikipedia.org/wiki/Solid-state_drive

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.19

SSD Architecture – Writes (II)

- Write A, B, C, D

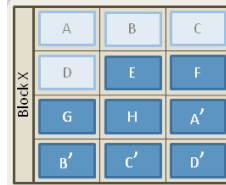
	A	B	C
Block X	D	free	free
	free	free	free
	free	free	free

https://en.wikipedia.org/wiki/Solid-state_drive

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.20

SSD Architecture – Writes (II)

- Write A, B, C, D
- Write E, F, G, H and A', B', C', D'
 - Record A, B, C, D as obsolete

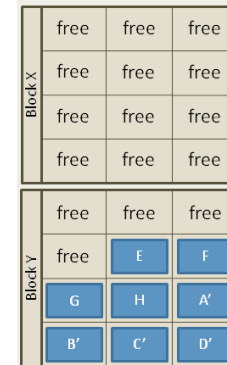


https://en.wikipedia.org/wiki/Solid-state_drive

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.21

SSD Architecture – Writes (II)

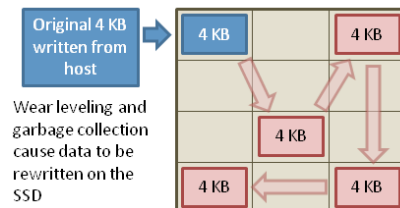
- Write A, B, C, D
- Write E, F, G, H and A', B', C', D'
 - Record A, B, C, D as obsolete
- Controller *garbage collects* obsolete pages by copying valid pages to new block
 - One erase every 64 or 128 writes



10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.22

SSD Architecture – Writes (III)

- Write and erase cycles require “high” voltage
 - Damages memory cells, limits SSD lifespan
 - Controller uses ECC, performs wear leveling



- Result is very workload dependent performance
 - Latency = Queuing Time + Controller time (Find Free Block) + Xfer Time
 - Highest BW: Seq. OR Random writes (limited by empty pages)

Rule of thumb: writes 10x more expensive than reads, and erases 10x more expensive than writes

Storage Performance & Price

	Bandwidth (sequential R/W)	Cost/GB	Size
HDD	50-100 MB/s	\$0.05-0.1/GB	2-4 TB
SSD ¹	200-550 MB/s (SATA) 6 GB/s (read PCI) 4.4 GB/s (write PCI)	\$1-1.5/GB	200GB-1TB
DRAM	10-16 GB/s	\$5-10/GB	64GB-256GB

¹<http://www.fastestssd.com/featured/ssd-rankings-the-fastest-solid-state-drives/>

BW: SSD up to x10 than HDD, DRAM > x10 than SSD
Price: HDD x20 less than SSD, SSD x5 less than DRAM

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.24

SSD Summary

- Pros (vs. hard disk drives):
 - Low latency, high throughput (eliminate seek/rotational delay)
 - No moving parts:
 - » Very light weight, low power, silent, very shock insensitive
 - Read at memory speeds (limited by controller and I/O bus)
- Cons
 - Small storage (0.1-0.5x disk), very expensive (20x disk)
 - » Hybrid alternative: combine small SSD with large HDD
 - Asymmetric block write performance: read pg/erase/write pg
 - » Controller garbage collection (GC) algorithms have major effect on performance
 - Limited drive lifetime
 - » 1-10K writes/page for MLC NAND
 - » Avg failure rate is 6 years, life expectancy is 9–11 years

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.25

Administrivia

- Project 2 Design Doc due Thursday 10/17 at 11:59PM
- Midterm #1 is Monday Oct 21 5:30-7pm in **145 Dwinelle (A-L)** and **2060 Valley LSB (M-Z)**
 - Closed book, ONE double-sided **handwritten** page of notes, no calculators, smartphones, Google glass etc.
 - Covers lectures #1-13 (today), readings, handouts, and projects 1 and 2
 - Review session **390 Hearst Mining, Fri October 18, 5-7 PM**
- Please fill out the anonymous course survey at <https://www.surveymonkey.com/s/FSW3HVJ>
 - We'll try to make changes *this* semester based on your feedback

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.26

Quiz 13.2: HDDs and SSDs

- Q1: True _ False _ The block is the smallest addressable unit on a disk
- Q2: True _ False _ An SSD has zero seek time
- Q3: True _ False _ For an HDD, the read and write latencies are similar
- Q4: True _ False _ For an SSD, the read and write latencies are similar
- Q5: Consider the following sequence of requests (2, 4, 1, 8), and assume the head position is on track 9. Then, the order in which SSTF services the requests is _____

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.27

5min Break

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.28

Quiz 13.2: HDDs and SSDs

- Q1: True _ False **x** The block is the smallest addressable unit on a disk
- Q2: True **x** False _ An SSD has zero seek time
- Q3: True **x** False _ For an HDD, the read and write latencies are similar
- Q4: True _ False **x** For an SSD, the read and write latencies are similar
- Q5: Consider the following sequence of requests (2, 4, 1, 8), and assume the head position is on track 9. Then, the order in which SSTF services the requests is (8, 4, 2, 1)

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.29

Building a File System

- **File System:** Layer of OS that transforms block interface of disks (or other block devices) into Files, Directories, etc.
- File System Components
 - Disk Management: organizing disk blocks into files
 - Naming: Interface to find files by name, not by blocks
 - Protection: Layers to keep data secure
 - Reliability/Durability: Keeping of files durable despite crashes, media failures, attacks, etc.

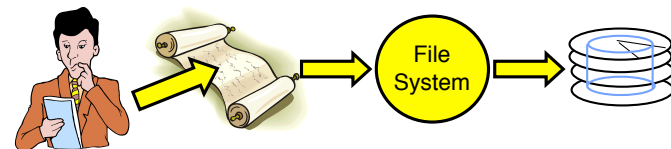
10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.30

User vs. System View of a File

- User's view:
 - Durable Data Structures
- System's view (system call interface):
 - Collection of Bytes (UNIX)
 - Doesn't matter to system what kind of data structures you want to store on disk!
- System's view (inside OS):
 - Collection of blocks (a block is a logical transfer unit, while a sector is the physical transfer unit)
 - Block size \geq sector size; in UNIX, block size is 4KB

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.31

Translating from User to System View



- What happens if user says: give me bytes 2—12?
 - Fetch block corresponding to those bytes
 - Return just the correct portion of the block
- What about: write bytes 2—12?
 - Fetch block
 - Modify portion
 - Write out Block
- Everything inside File System is in whole size blocks
 - For example, `getc()`, `putc()` \Rightarrow buffers something like 4096 bytes, even if interface is one byte at a time
- From now on, file is a collection of blocks

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.32

Disk Management Policies

- Basic entities on a disk:
 - **File**: user-visible group of blocks arranged sequentially in logical space
 - **Directory**: user-visible mapping of names to files
- Access disk as linear array of sectors.
 - **Logical Block Addressing (LBA)**: Every sector has integer address from zero up to max number of sectors
 - » Controller must deal with bad sectors (formerly OS/BIOS)
 - Controller translates from address \Rightarrow physical position
 - » Hardware shields OS from structure of disk

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.33

Disk Management Policies (cont'd)

- Need way to track free disk blocks
 - Link free blocks together \Rightarrow too slow today
 - Use bitmap to represent free space on disk
- Need way to structure files: **File Header**
 - Track which blocks belong at which offsets within the logical file structure
- **Optimize placement of files' disk blocks to match access and usage patterns**

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.34

Designing the File System: Access Patterns

- Sequential Access: bytes read in order (“give me the next X bytes, then give me next, etc.”)
 - Most of file accesses are of this flavor
- Random Access: read/write element out of middle of array (“give me bytes $i-j$ ”)
 - Less frequent, but still important, e.g., mem. page from swap file
 - Want this to be fast – don't want to have to read all bytes to get to the middle of the file
- Content-based Access: (“find me 100 bytes starting with JOSEPH”)
 - Example: employee records – once you find the bytes, increase my salary by a factor of 2
 - Many systems don't provide this; instead, build DBs on top of disk access to index content (requires efficient random access)
 - Example: Mac OSX Spotlight search (do we need directories?)

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.35

Designing the File System: Usage Patterns

- Most files are small (for example, .login, .c, .java files)
 - A few files are big – executables, swap, .jar, core files, etc.; the .jar is as big as all of your .class files combined
 - However, most files are small – .class, .o, .c, .doc, .txt, etc
- Large files use up most of the disk space and bandwidth to/from disk
 - May seem contradictory, but a few enormous files are equivalent to an immense # of small files
- Although we will use these observations, beware!
 - Good idea to look at usage patterns: beat competitors by optimizing for frequent patterns
 - Except: changes in performance or cost can alter usage patterns. Maybe UNIX has lots of small files because big files are really inefficient?

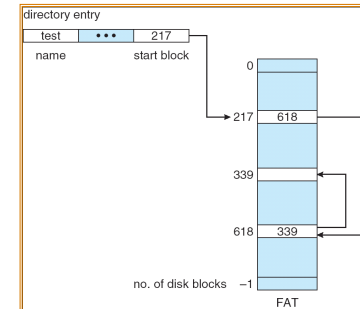
10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.36

File System Goals

- Maximize sequential performance
- Efficient random access to file
- Easy management of files (growth, truncation, etc)

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.37

Linked Allocation: File-Allocation Table (FAT)



- MSDOS links pages together to create a file
 - Links not in pages, but in the File Allocation Table (FAT)
 - » FAT contains an entry for each block on the disk
 - » FAT Entries corresponding to blocks of file linked together
 - Access properties:
 - » Sequential access expensive unless FAT cached in memory
 - » Random access expensive always, but *really* expensive if FAT not cached in memory

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.38

Quiz 13.3: Deadlocks

- Q1: True _ False _ If a resource type (e.g., disk) has multiple instances we cannot have deadlock
- Q2: True _ False _ Deadlock implies starvation
- Q3: True _ False _ Starvation implies deadlock
- Q4: True _ False _ If resources can be preempted from threads we cannot have deadlock
- Q5: True _ False _ Assume a system in which each thread is only allowed to either allocate all resources it needs or none of them. In such a system we can still have deadlock.

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.39

Quiz 13.3: Deadlocks

- Q1: True _ False X If a resource type (e.g., disk) has multiple instances we cannot have deadlock
- Q2: True X False _ Deadlock implies starvation
- Q3: True _ False X Starvation implies deadlock
- Q4: True X False _ If resources can be preempted from threads we cannot have deadlock
- Q5: True _ False X Assume a system in which each thread is only allowed to either allocate all resources it needs or none of them. In such a system we can still have deadlock.

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.40

Summary (1/2)

- Hard (Magnetic) Disk Performance:
 - Latency = Queuing time + Controller + Seek + Rotational + Transfer
 - Rotational latency: on average $\frac{1}{2}$ rotation
 - Transfer time: depends on rotation speed and bit density
- SSD Performance:
 - Read: Queuing time + Controller + Transfer
 - Write: Queuing time + Controller (Find Free Block) + Transfer
 - Find Free Block time: depends on how full SSD is (available empty pages), write burst duration, ...
 - Limited drive lifespan

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.41

Summary (2/2)

- File System:
 - Transforms blocks into Files and Directories
 - Optimize for access and usage patterns
 - Maximize sequential access, allow efficient random access
- File (and directory) defined by header, called “inode”

10/16/2013 Anthony D. Joseph and John Canny CS162 ©UCB Fall 2013 13.42