Operating Systems Design (CS 423)

Elsa L Gunter
2112 SC, UIUC
http://www.cs.illinois.edu/class/cs423/

Based on slides by Roy Campbell, Sam King, and Andrew S Tanenbaum

File Systems Support

- Files Systems – abstraction of stored data
- Data stored on device
- Accessed through device drive
- Nature of file system heavily influence by nature of storage device

Device Driver Abstractions

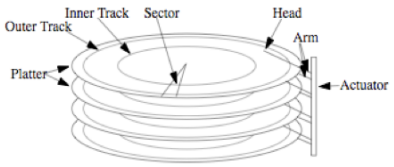
- OS Abstractions make things "better"
  - Threads: don't worry about sharing CPU
  - Address spaces: don't worry about sharing physical memory
  - Device driver: don't worry about differences and limitations of devices

Byte-oriented access vs block oriented

- Disks are accessed in terms of blocks (sectors) of 512 bytes
- Programs deal in bytes, notblocks
- How can you read less than a block?
- How can you write less than a block?

Disk geometry and access

- Disk made of stack of spinning **platters**
- Top and bottom, concentric circles of data (**tracks),** tracks at same radial distance are called **cylinders**
- Each track has number of **sectors**



Seagate Barracuda 1 TB Int hard drive

Accessing a Disk

- Queuing time (wait for it to be free) 0-∞
- Position disk arm and head (seek and rotate) 0-12ms
- Access disk data:
  - typical 50-70MB/s
  - Best currently: 300MB/s
- Increased disk rotations means speeds generally faster
- Faster access times for outside tracks

Optimizing Disk Performance

- Disk is slow!
- Best option: caching to eliminate I/O
- When you go to disk, keep positioning time low
- If you do have to re-position, try to amortize

Reducing Positioning Time

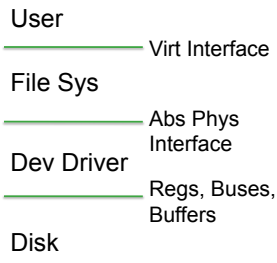
- Optimize when writing data to disk
  - Place items that will be accessed together near each other on disk
  - How do you know in advanced that two items will be accessed together?
- Optimize when reading data from disk
  - Minimize positioning time at time of access, rather than at time of creation
  - Disk scheduling–will not talk about in this class

Solid state disks

- Remove mechanical components
- Advantages
  - Fast!
  - Apparently can saturate SATA (1.5Gb/s)
  - No seek time, waiting for spinning
  - More reliable (maybe)
- Disadvantages
  - More expensive
  - Limited write cycles
    - Write leveling helps
- MacBook Pro with 256GB SSD
- Vista–fast boot optionusing USB flashdrive
- Hybrid SSD/traditional disk

File Systems

- File system: OS abstraction to make disk easier to use
- Physical reality
  - Slow access to disk blocks
- Illusion provided
  - Fast access to byte oriented files, indexed using symbolic (user-spec) names



The File Illusion: How to Implement?

- How to map file space onto disk space?
  - File system structure on disk; disk allocation
  - Very similar to memory management
- How to use symbolic names instead of disk sectors?
  - Naming; directories
  - Not similar to memory management where virtual and physical both use same name (i.e.address)

File System Structure

- Overall question: how to organize files on disk
  - What data structure is right one to use?
  - Side note: many things in OS (and CS in general) boil down to data structures and algorithms

File System Structure

- Need internal structure to describe object
  - Called "file header" in this class
    - Inode in Unix
  - File header also contains miscellaneous information about file, e.g., file size, modification date, permissions
    - Also called file meta-data
- Many ways to organize data on disk

File System Usage Patterns

- 80% of file accesses are reads
- Most programs that access file sequentially access the entire file
  - Alternative is random access
    - Examples?
- Most files are small; most bytes on disk are from large files

Contiguous Allocation

- Store file in one contiguous segment on disk (sometimes called an extent)
  - User must declare size of file in advance
  - File system will pre-allocate this memory on disk
- What do you do if file grows larger?
- File header is simple: starting block num & size
- Similar to base & bounds for mem mngt

Contiguous Allocation

- Pros
  - No seeks between blocks
  - Easy random access
  - Easy and fast to calculate any block in file
- Cons
  - External fragmentation
  - Hard to grow files
  - Wastes space

Linked List

- Each block contains pointer to next block of file (along with data)
  - Used by Alto (first personal computer)
- File header contains pointer to first disk block

Linked List

- Pros
  - Grow easily (i.e. append) files
  - No external fragmentation (pick any free block)
- Cons
  - Sequential access quite slow
  - Lots of seeks between blocks
  - Random access is really slow