

## Disk Storage and File Systems

CS 256/456  
Dept. of Computer Science, University  
of Rochester

10/30/2008

CSC 2/456

1

## Recap of the Last Class: Disk Storage

- Disk drive
  - mechanical parts (cylinders, tracks, sectors) and how they move to access disk data
  - electronic part (disk controller main) exposes an one-dimensionally addressable set of blocks
  - large seek/rotation time
- Disk scheduling
  - goals: overall efficiency; fairness (no starvation)
  - FCFS, SSTF, SCAN, C-SCAN, LOOK

10/30/2008

CSC 2/456

2

## Deadline Scheduling in Linux

- A regular elevator-style scheduler similar to C-LOOK
- Additionally, all I/O requests are put into a FIFO queue with an expiration time (e.g., 500ms)
- When the head request in the FIFO queue expires, it will be executed next (even if it is not next in line according to C-LOOK).
- A mix of performance and fairness.

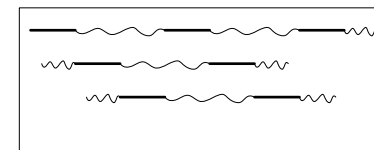
10/30/2008

CSC 2/456

3

## Concurrent I/O

- Consider two request handlers in a Web server
  - each accesses a different stream of sequential data (a file) on disk;
  - each reads a chunk (the buffer size) at a time; does a little CPU processing; and reads the next chunk
- What happens?



10/30/2008

CSC 2/456

4

## How to Deal with It?

- Aggressive prefetching
- Anticipatory scheduling [Iyer & Druschel, SOSP 2001]
  - at the completion of an I/O request, the disk scheduler will wait a bit (despite the fact that there is other work to do), in anticipation that a new request with strong locality will be issued; schedule another request if no such new request appears before timeout
  - included in Linux 2.6

10/30/2008

CSC 2/456

5

## Exploiting Concurrency

- RAID: Redundant Arrays of Independent Disks
  - RAID 0: data striping at block level, no redundancy
  - RAID 1: mirrored disks (100% overhead)
  - RAID 2: bit-level striping with parity bits, synchronized writes
  - RAID 3: data striping at the bit level with parity disk, synchronized writes
  - RAID 4: data striping at block level with parity disk
  - RAID 5: scattered parity
  - RAID 6: handles multiple disk failures

10/30/2008

CSC 2/456

6

## Disk Management

- Formatting
  - Header: sector number etc.
  - Footer/tail: ECC codes
  - Gap
  - Initialize mapping from logical block number to defect-free sectors
- Logical disk partitioning
  - One or more groups of cylinders
  - Sector 0: master boot record loaded by BIOS firmware, which contains partition information
  - Boot record points to boot partition

10/30/2008

CSC 2/456

7

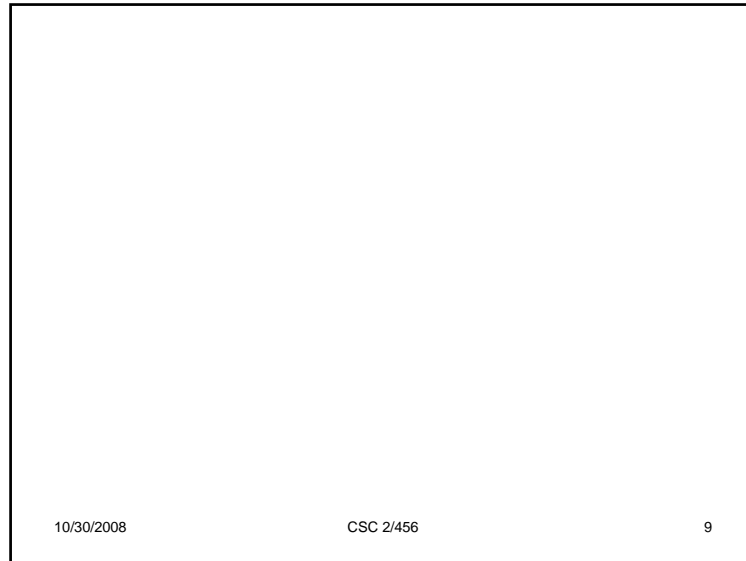
## Swap Space Management

- Part of file system?
  - Requires navigating directory structure
  - Disk allocation data structures
- Separate disk partition
  - No file system or directory structure
  - Optimize for speed rather than storage efficiency
  - When is swap space created?

10/30/2008

CSC 2/456

8



10/30/2008

CSC 2/456

9

## File Systems

- A File system is the OS abstraction for storage resources
  - File is a logical storage unit in the OS abstract interface for storage resources
    - Extension of address space (temporary files)
    - Non-volatile storage that survives the execution of an individual program (persistent files)
  - Directory is a logical "container" for a group of files

10/30/2008

CSC 2/456

10

## Operations Supported

- Create – associate a name with a file
- Delete – remove the file
- Rename – associate a new name with a file
- Open – create cached context that is associated implicitly with future reads and writes
- Write – store data in a file
- Read – access the data associated with a file
- Close – discard cached context
- Seek – random access to any record or byte
- Map – place in address space for convenience (memory-based loads and stores), speed; disadvantages: lengths that are not multiples of the page size, consistency with open/read/write interface

10/30/2008

CSC 2/456

11

## File System Issues

- File naming and other attributes:
  - name, size, access time, sharing/protection, location
- Intra-file structure
  - None - sequence of words, bytes
  - Complex Structures
    - records/formatted document/executable
- File system organization: efficiency of disk access
- Concurrent access: allow multiple processes to read/write
- Reliability: integrity in the presence of failures
- Protection: sharing/protection attributes and access control lists (ACLs)

10/30/2008

CSC 2/456

12

## File Naming

- Fixed vs. variable length
  - Fixed: 8-255 characters
  - Variable: length:value encoding
- File extensions – system supported vs. convention

10/30/2008

CSC 2/456

13

## Naming Files Using Directory Structures

- Directory: maps names to files; directories may themselves be files
  - Single level (flat): no two files may have the same name
  - Two level: per-user single-level directory
  - Hierarchical: generalization of two level; each file system is assigned the root of a tree
  - Acyclic (or cyclic) graph: allow sharing of files across directories; hard versus soft (symbolic) links

10/30/2008

CSC 2/456

14

## Shared Files: Links

- File appears simultaneously in different directories
- File system is now a directed acyclic graph (DAG)
- **Hard** link – directory points to file inode, which maintains a count of pointers
- **Soft** link – new file type, containing the path of the file to which it is linked, along with permissions (symbolic linking) – no pointer to inode

10/30/2008

CSC 2/456

15

## File Types

- Control operations allowed on files
- Use file name extensions to indicate type (in Unix, this is just a convention)
- Structured vs. unstructured data
  - None - sequence of words, bytes
  - Complex Structures
    - records/formatted document/executable
- Sequential, random, or key-based (indexed) access

10/30/2008

CSC 2/456

16

## File Space Organization

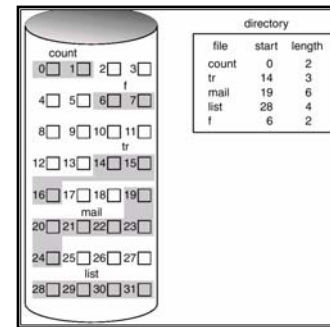
- Disk basic allocation unit is a sector (e.g., 512 bytes)
- File system may choose to use a larger block size (e.g., 4KB)
- File allocation methods
  - How disk blocks are allocated for files
    - Contiguous allocation
    - Linked allocation
    - Indexed allocation
  - Metrics:
    - Access speed (sequential & random)
    - Space utilization

10/30/2008

CSC 2/456

17

## Contiguous File Allocation



- Each file occupies a set of contiguous blocks on the disk
- Advantage:
  - Simple - only starting location (block #) and length (number of blocks) are required
  - Fast sequential; also quite fast random access
- Disadvantage:
  - External fragmentation
  - Inflexible when appending to a file

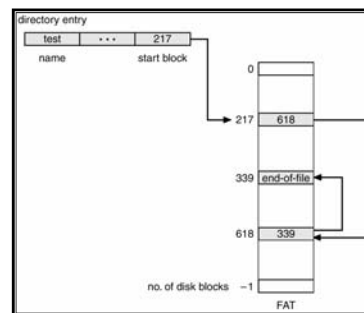
10/30/2008

CSC 2/456

18

## Linked File Allocation

- Each file is a linked list of disk blocks
  - each block contains a next pointer
  - directory only needs to store the pointer to the first block
  - blocks may be scattered anywhere on the disk
- Advantage
  - Space efficient
  - Flexible in appending
- Disadvantage:
  - Poor access speed (sequential & random)



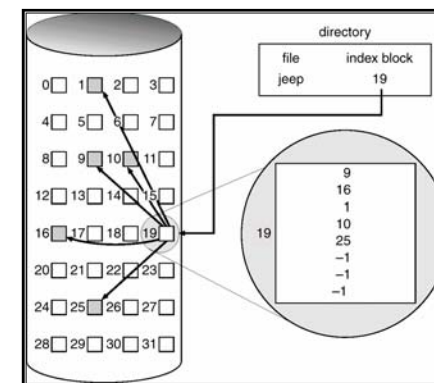
10/30/2008

CSC 2/456

19

## Indexed File Allocation

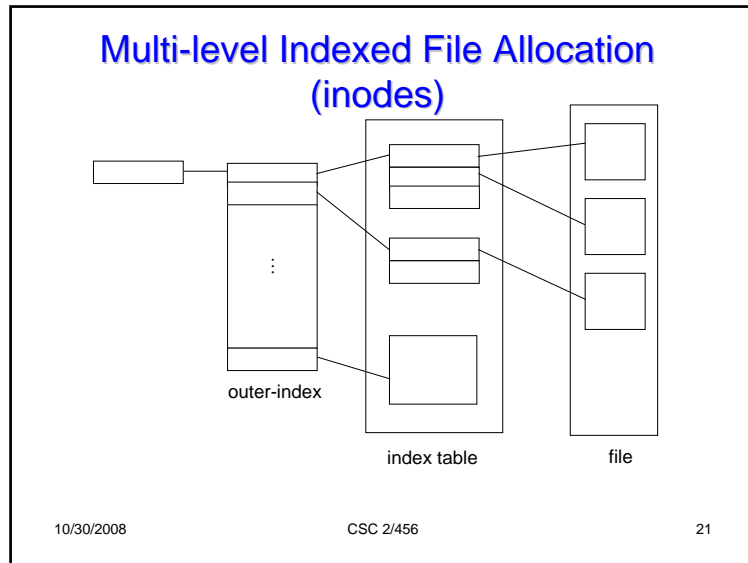
- Brings all pointers together into the *index block*.



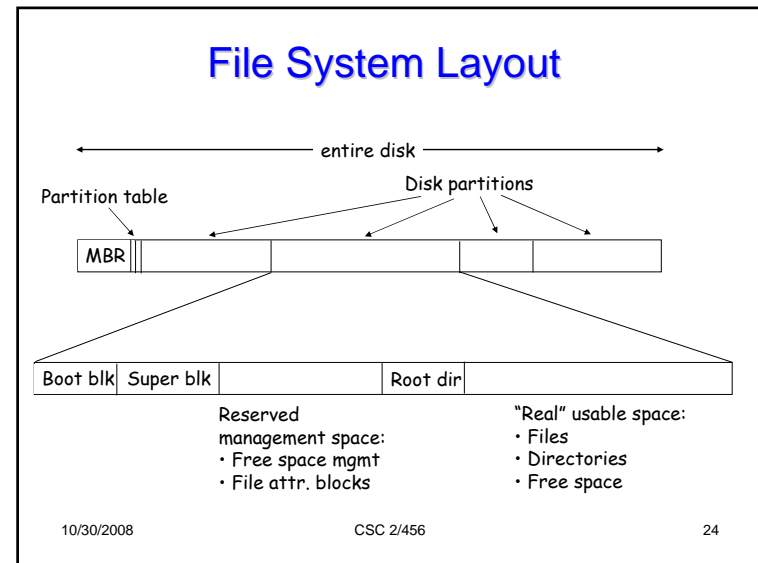
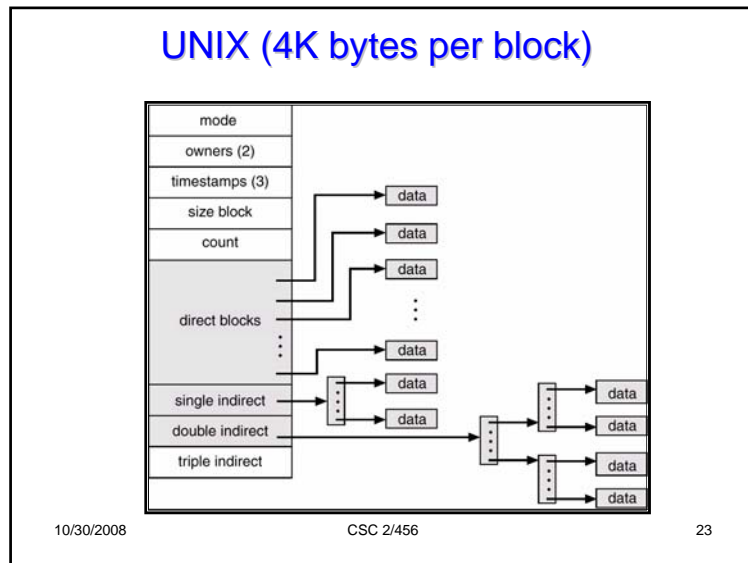
10/30/2008

CSC 2/456

20



- ### Indexed Allocation (pros and cons)
- Space efficiency
    - no external fragmentation
    - overhead of index blocks
  
  - Access speed
    - random access
    - sequential access
- 10/30/2008                  CSC 2/456                  22



## In-Memory Structures

- Used for file system management and performance improvement via caching
  - Mount table (info on each mounted volume)
  - Directory-structure cache
  - System-wide open file table
    - Copy of FCB (file control block) of each open file
  - Per-process open file table
    - Pointer to entry in system-wide table along with process-specific information
- Open system call returns a pointer to the appropriate entry in per-process file table (file descriptor or file handle)

10/30/2008

CSC 2/456

25

## Directory on the Disk

- Directory is a container of files
- For space management, similar to files
- But for directory, file system does care about its content
  - Linear list of file names and attributes (including pointers to the data blocks)
    - time-consuming to search an item
  - Hash Table - using a link list to chain all files hashed to the same value
    - Pro: decreases directory search time
    - Con: increased complexity, a little waste of space
    - how much benefit does it really provide?

10/30/2008

CSC 2/456

26

## Where to put file attributes?

- File control block - data structure including all attributes for a file
- Where to put the file control block?
  - In the directory data structure
    - Hard to share files through links
  - In the system-level dedicated data structure
    - inode

10/30/2008

CSC 2/456

27

## File Sharing and Protection

- Sharing of files on multi-user systems is desirable
- Sharing must be accompanied by a *protection* scheme
  - In general, a protection scheme specifies whether any specific user can access any specific file
    - Access control lists (ACL)
    - User, group, other permissions

10/30/2008

CSC 2/456

28

## Device Space Management

- Block size: internal fragmentation/wasted space vs. allocation efficiency and access latency
- Free space management
- Reducing disk arm motion

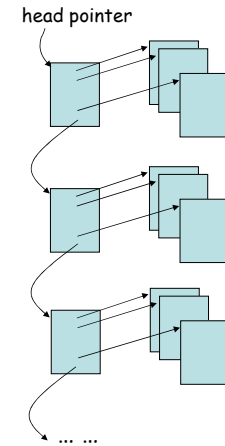
10/30/2008

CSC 2/456

29

## Free-Space Management

- Free-space management for memory
- Bit map and linked free block list
- Space overhead: bit vs. word
- Efficiency
  - getting the address of one free block
  - getting the addresses of a number of free blocks
- Alternative: Grouping/clustering



10/30/2008

CSC 2/456

30

## Delayed Writes and Data Loss at Machine Crash

- Writes are commonly delayed for better performance
  - data to be written is cached
- A sudden machine crash may result in a loss of data
  - a completed write does not mean the data is safely stored on storage
- fsync() - flush all delayed writes to disk
  - fsync() may not even be totally safe with delayed writes on disk controller buffer cache

10/30/2008

CSC 2/456

31

## Consistency: Weaker Form of Reliability

- File system operations are not atomic; a sudden machine crash may leave the file system in an inconsistent state
- Consistency checking and fix (fsck, scandisk)
  - it takes too long
- Journaling file system:
  - maintain a dedicated journal that logs all operations
  - the logging happens before the real operation
  - each logging is made to be atomic
  - after the completion of an operation, its entry is removed from the journal
  - at the recovery time, only journal entries need to be examined ⇒ fast recovery
  - similar to transactions in database systems
  - performance impact?

10/30/2008

CSC 2/456

32

## Log-Structured File Systems

- With CPUs faster, memory larger
  - buffer caches can also be larger
  - most of read requests can come from the memory cache
  - thus, most disk accesses will be writes
  - poor disk performance when most writes are small
- LFS Strategy [Rosenblum&Ousterhout SOSP1991]
  - structures entire disk as a log
  - always write to the end of the disk log
  - when updates are needed, simply add new copies with updated content; old copies of the blocks are still in the earlier portion of the log
  - periodically purge out useless blocks

10/30/2008

CSC 2/456

33

## NFS (Network File System)

- Deals with heterogeneity using RPC/XDR
- Stateless – no open and close of files
- Interface transparent to user via VFS

10/30/2008

CSC 2/456

34

## Example File Systems

- MS-DOS/Windows – file allocation table (FAT), NTFS
- Linux – VFS, ext2fs
- NFS
- ...

10/30/2008

CSC 2/456

35

## File System Caching

- File content is cached in memory buffer for later reuse
  - what is the basic unit of such caching?
    - Disk blocks vs. clusters vs. pages
- Replacement policy for file system buffer cache
  - LRU replacement is one possibility; but sequential access is very likely in file system I/O
  - MRU or free-behind

10/30/2008

CSC 2/456

36

## File System Prefetching

- File content is read ahead of time for anticipated use in the near future
- Often sequential (based on past access history on the file)
- What is the advantage of file prefetching?
- What is the danger of file prefetching?
- A balanced scheme that provides a competitive performance to the optimal scheme [Li et al. EuroSys 2007]

10/30/2008

CSC 2/456

37

## Informed Prefetching

- Informed prefetching - prefetching while utilizing some information about application data access pattern
- Application I/O hints [Cao et al. 1994] [Patterson et al. 1995]
- Automatic I/O hints based on speculative execution [Chang&Gibson 2000], [Fraser&Chang 2003]

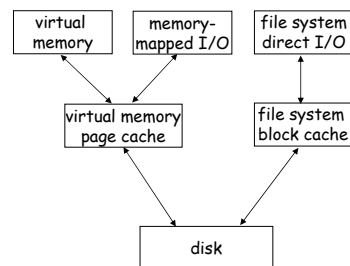
10/30/2008

CSC 2/456

38

## Buffer Cache in Main Memory

- Memory-mapped I/O naturally share page cache with the virtual memory system
- Problems:
  - double buffering
  - inconsistencies



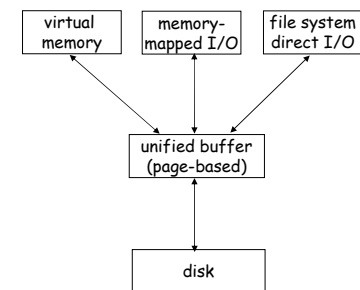
10/30/2008

CSC 2/456

39

## Unified Buffer Cache & Unified Virtual Memory

- A unified buffer cache uses the same page cache to store [Pai et al. 1999]
  - virtual memory pages
  - memory-mapped pages
  - file system direct I/O data



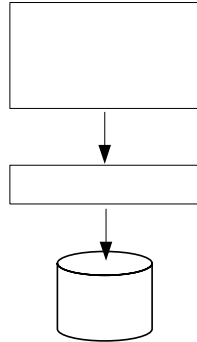
10/30/2008

CSC 2/456

40

## Multi-level I/O Buffer

- buffer cache in the main memory
- track cache on the disk controller



10/30/2008

CSC 2/456

41

## Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).

10/30/2008

CSC 2/456

42