File Systems

John Criswell
University of Rochester

Review of Disk I/O

Hard Disks

- Mechanical parts
  - Cylinders
  - Tracks
  - Sectors
- Electronic part
  - Disk controller exposes a one-dimensionally addressable set of blocks
- Large seek/rotation time

Disk Scheduling

- Disk scheduling – choose from outstanding disk requests when the disk is ready for a new request
  - Can be done in both disk controller and the operating system
  - Disk scheduling non-preemptible
- Goals of disk scheduling
  - overall efficiency – small resource consumption for completing disk I/O workload
  - fairness – prevent starvation

Special Blocks and Partitions

- 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

File Systems: Organizing the Chaos

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

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

File Abstraction

- File naming and other attributes:
  - name, size, access time, sharing/protection, location
- Internal file structure
  - No structure - sequence of words, bytes
  - Complex Structures
    - records/formatted document/executable

File System Issues

- File system organization: efficiency of disk access
- Concurrent access: allow multiple processes to read/write
- Reliability: integrity in the presence of failures
- Protection: who can perform which operations on files

File Naming

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

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

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

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

File System Organization

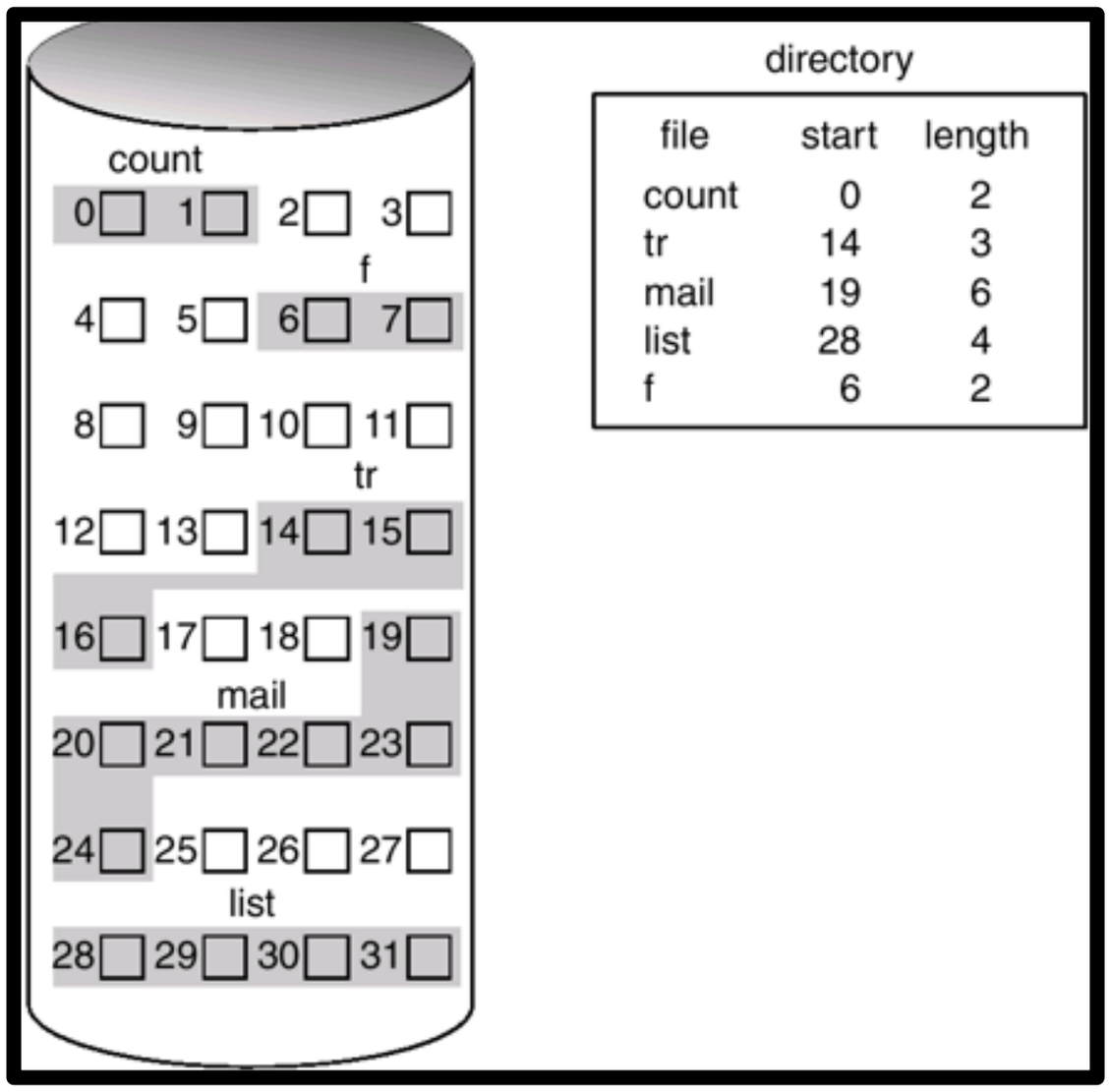
File System Block Size

- Disk basic allocation unit is a sector
  - (e.g., 512 bytes)
- File system may choose to use larger block size
  - (e.g., 4KB)

Disk Block Allocation Methods

- How disk blocks are allocated for files
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation
- Metrics:
  - Access speed (sequential & random)
  - Space utilization

Contiguous File Allocation

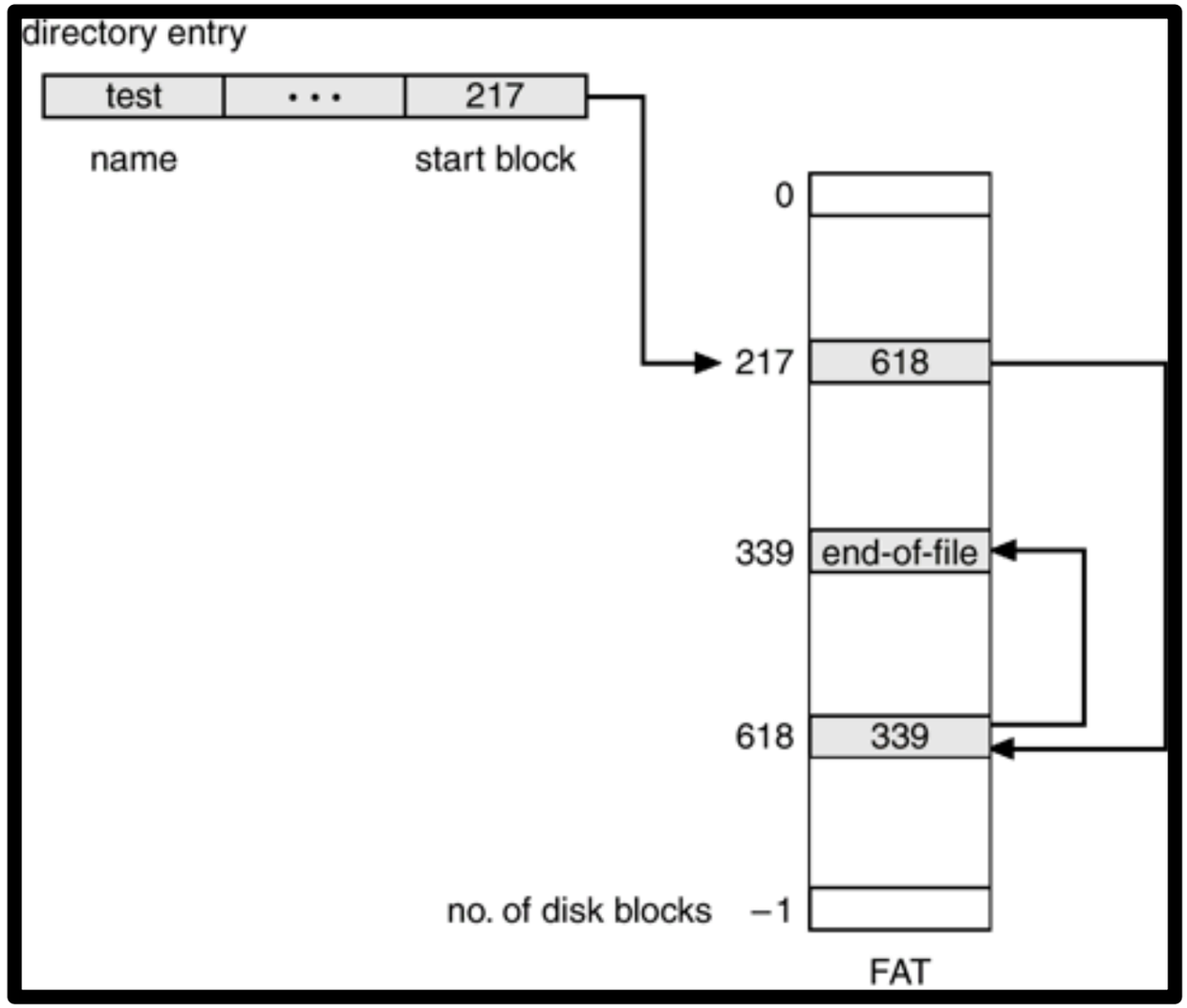


- Each file occupies a set of contiguous blocks on the disk
- Advantage:
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Fast sequential; also quite fast random access
- Disadvantage:
  - External fragmentation
  - Inflexible when appending to a file

When might you use contiguous file allocation?

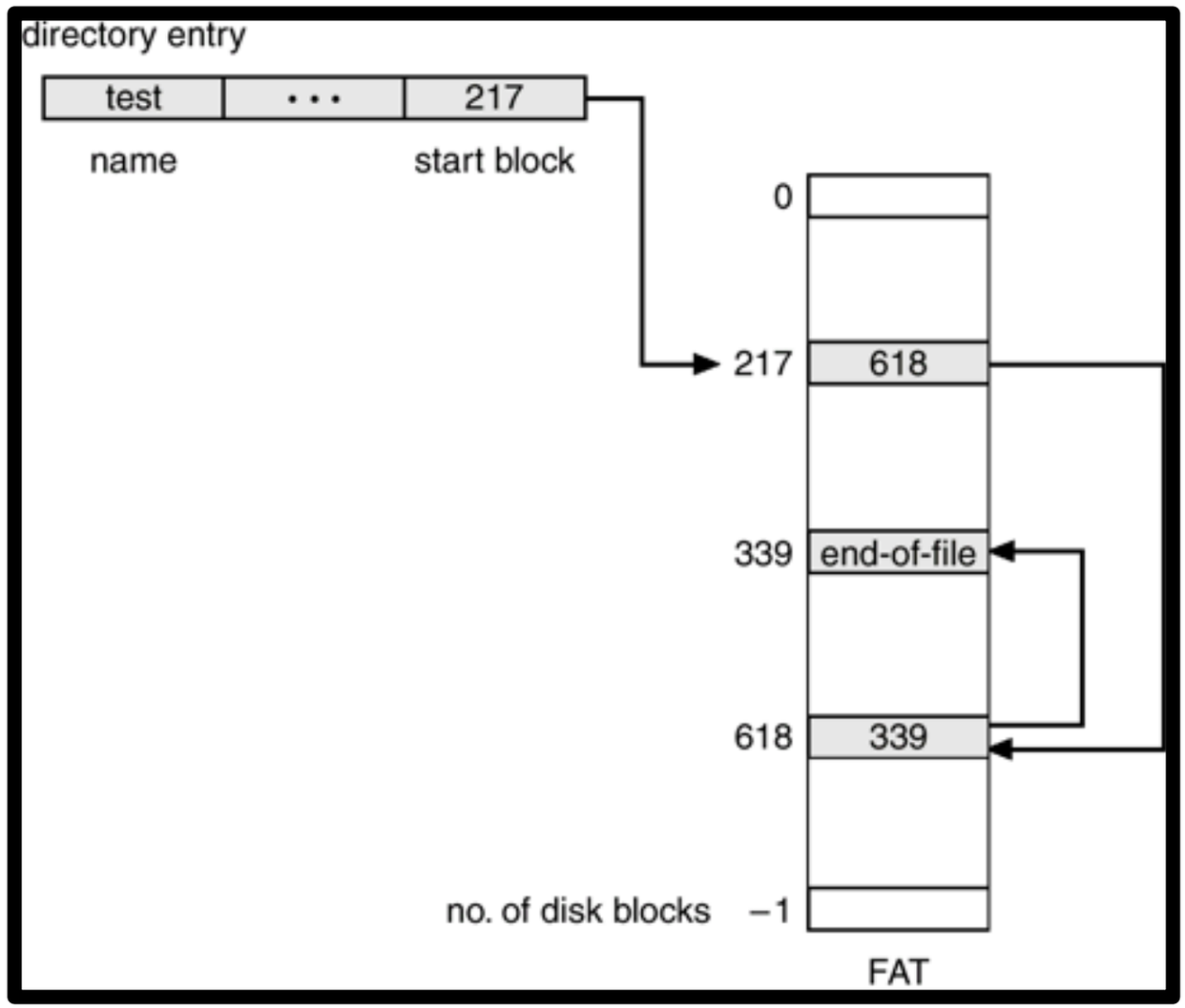
Linked File Allocation

- 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



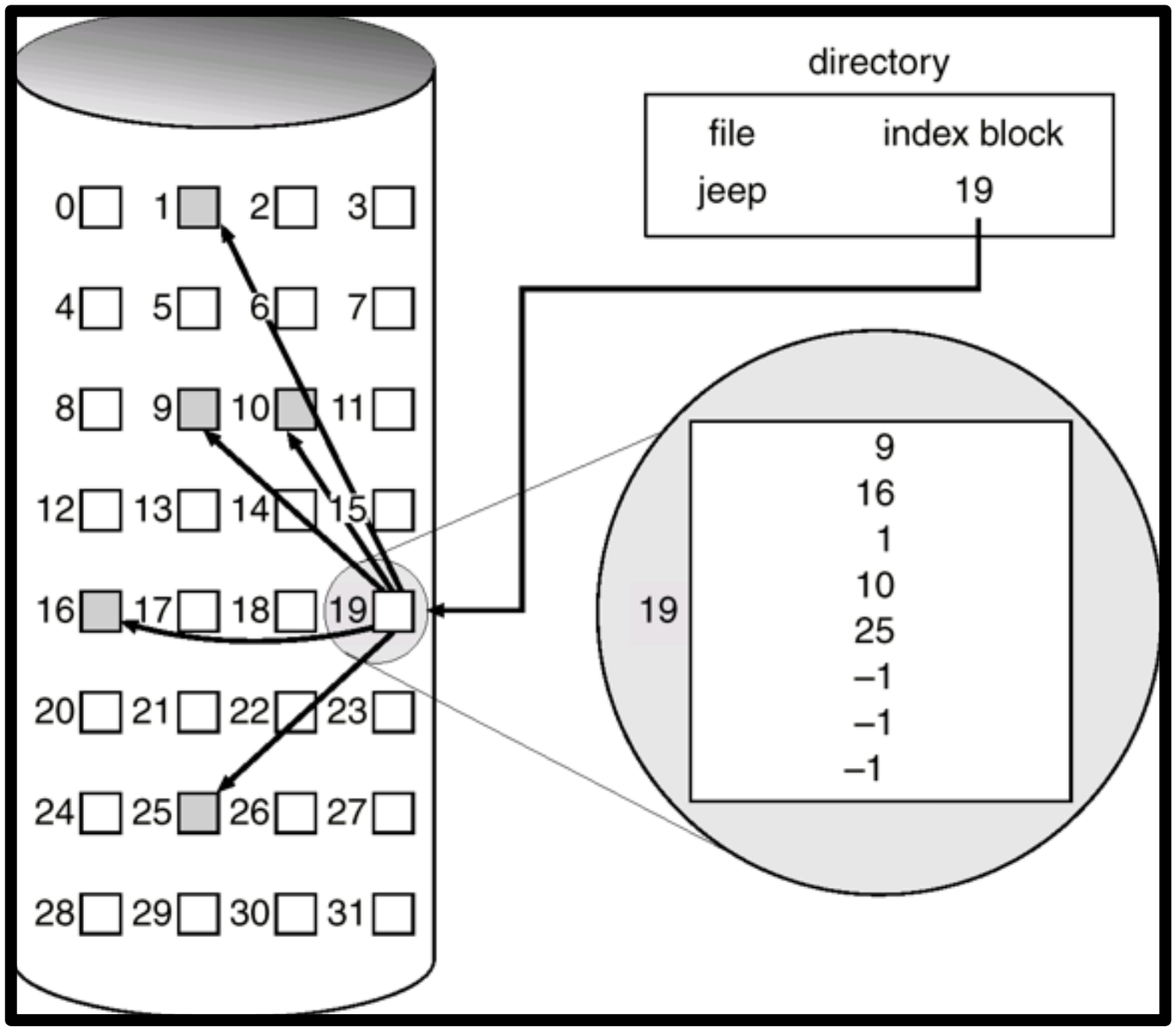
Linked File Allocation

- Advantage
  - Space efficient
  - Flexible in appending
- Disadvantage:
  - Poor access speed (sequential & random)



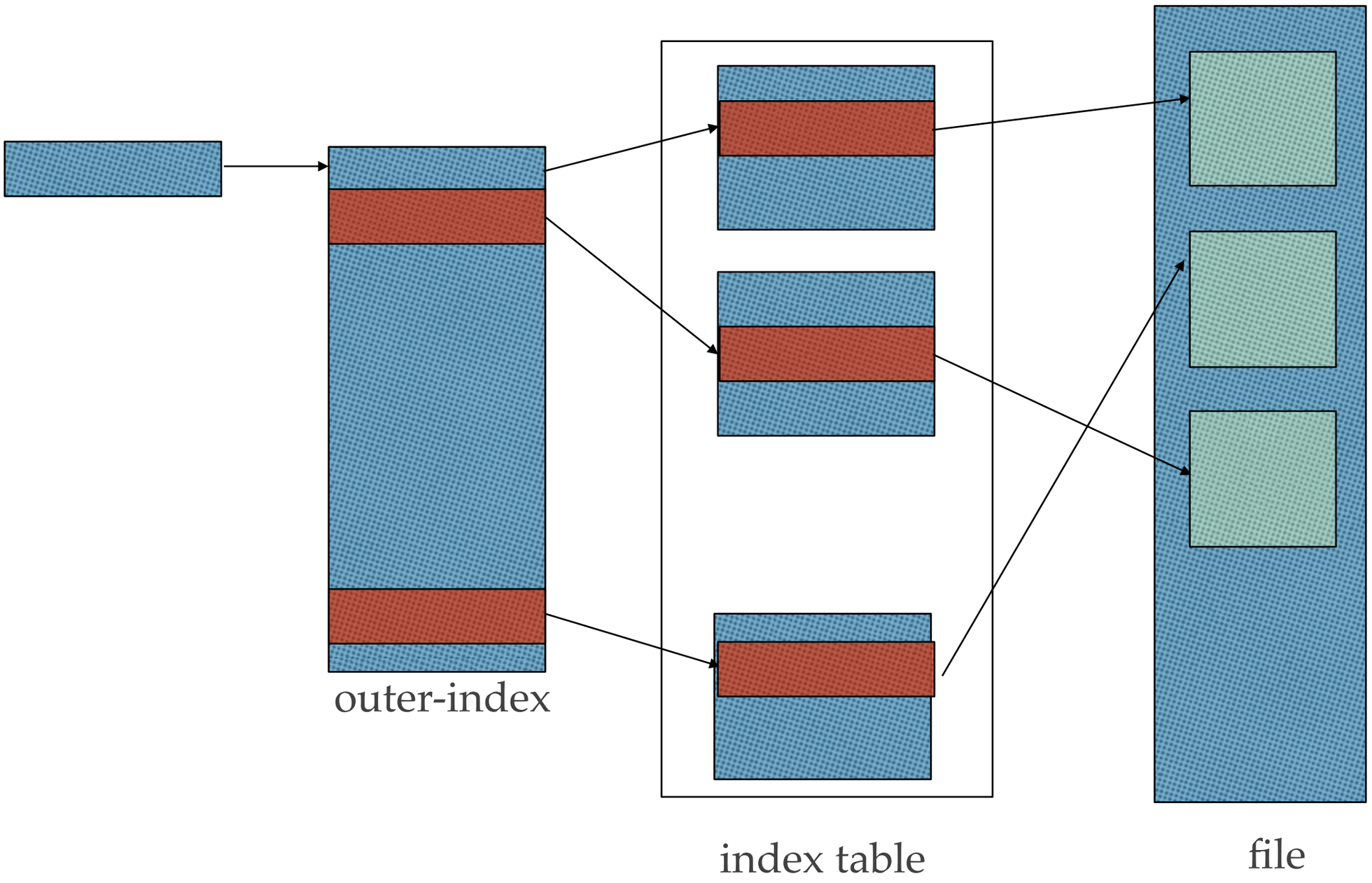
Indexed File Allocation

- Brings all pointers together into an index block



What is a limitation of using a single block for the block indices?

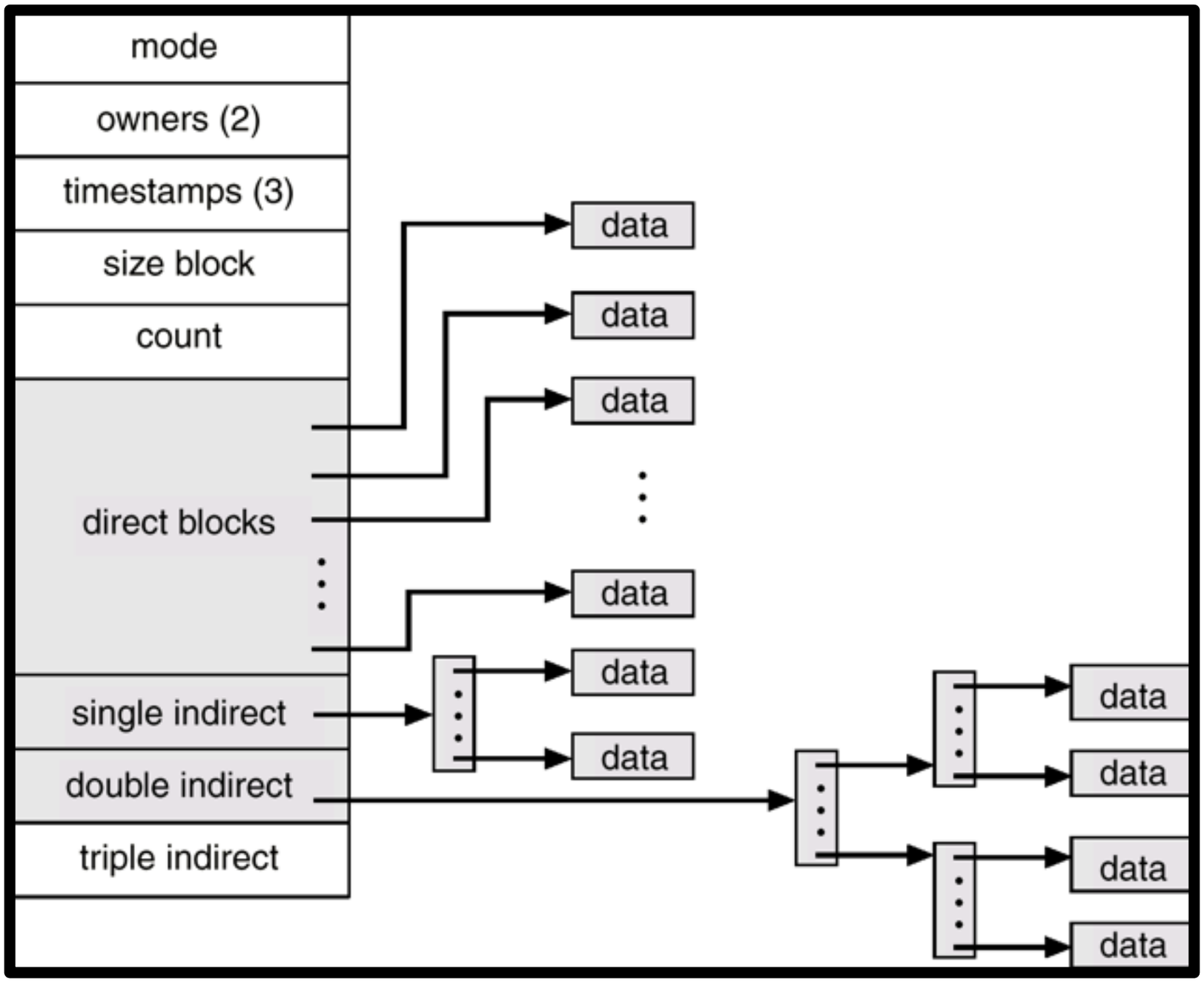
Multi-level Indexed File Allocation



Indexed Allocation (pros and cons)

- Space efficiency
  - no external fragmentation
  - overhead of index blocks
- Access speed
  - random access
  - sequential access

UNIX (4K bytes per block)



Where to Put Your Data Structures

In-Memory Structures

- Used for file system management and performance improvement via caching
  - Mount table (info on each mounted volume (aka “partition”))
  - 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)

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?

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

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

Device Space Management

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

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

Free-Space Management

Free-space management for memory

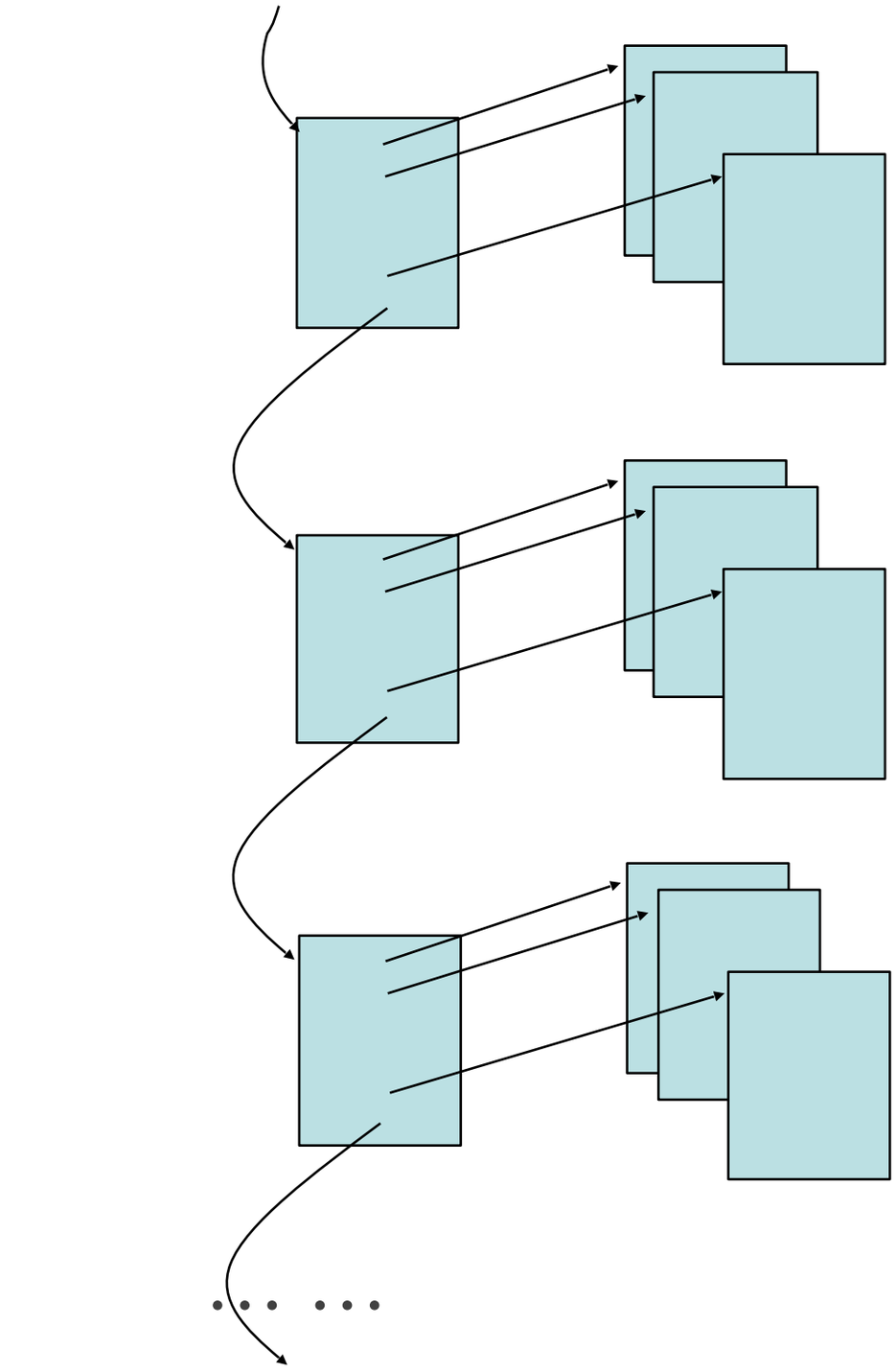
Bit map and linked free block list

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



Keeping Things in Memory

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

File System Caching: Buffer Cache

- 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

Consistency: Weaker Form of Reliability

- File system operations are not atomic; a sudden machine crash may leave the file system in an inconsistent state
- (In-)Consistency
  - Missing blocks
  - Duplicate free blocks
  - Duplicate data blocks
- Consistency checking and fix (fsck, scandisk)
  - use redundant data on disk to recover consistency
  - E.g., free block cannot be on the free list and in a file
  - Sometimes data is still lost

Join my petition!

Rename Fisk, WI to FSCK, WI!

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, SOSP 1991]
  - structure 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

"New" Motivations

- Fast recovery
  - Disks became *very big* (1 TB or more)
  - Consistency check (fsck/scandisk) too slow
- Persistency
  - Availability

Journaling

- 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

Journaling

- LFS is a dynamic journal
- Physical journal (ext3)
- Logical journal (NTFS)
- Snapshotting (ZFS)

Solid State Drives

- No mechanical component (moving parts)
- Lower energy requirements
- Speed
  - Reads and writes in the order of 10s of microseconds (reading faster than writing)
  - Erase on the order of a millisecond
- Finite number of erase and write cycles, requiring what is called “wear leveling”

Effect on File Systems

- No need to "cluster" data to reduce seek time
- Need to avoid writes to the same block
- File system cache less useful due to lower speed mismatch
- Log-structured file system for SSD
  - Provides wear leveling

Credits

- 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).