File System Implementation

Disks

- Usually, file systems are stored on hard drives.
  - Big, cheap, non-volatile, slow.
  - Organized into cylinders, tracks, sectors.
  - Those details are managed by a device driver.
    - File system implementation can work in terms of block numbers...
- Minimum unit of data access is a block.
  - Maybe 512 bytes.
- Moving the head is slow, but once the head is in place, transfer rate is high.

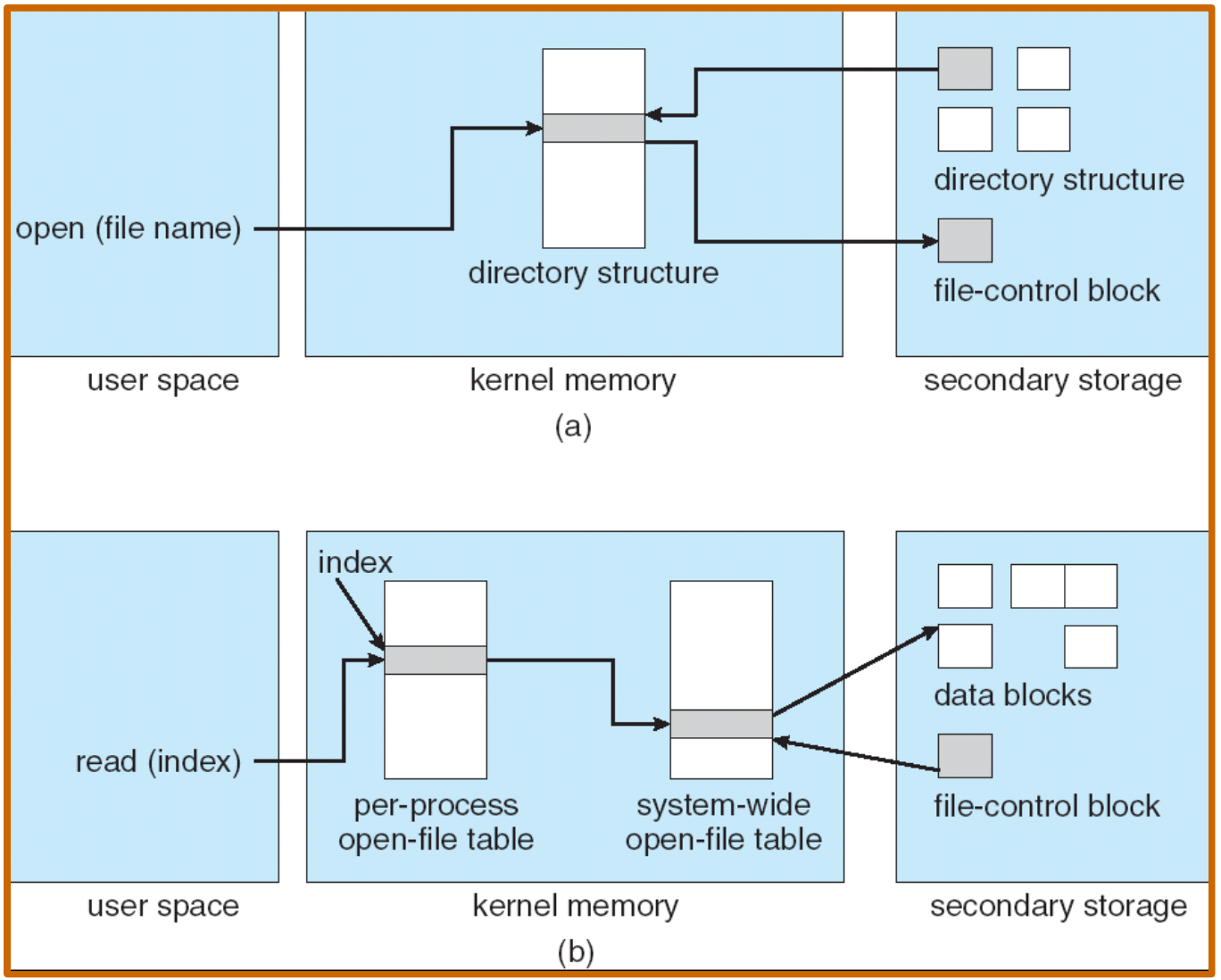
File System Structures

- Several types of structures are necessary to represent a file system.
  - These are often represented both as blocks on disk and OS data structures.
    - Boot control block.
    - Volume control block.
    - Directory structures.
    - File control blocks.
      - Permissions.
      - Access/Modify times.
      - Owner.
      - Size.
      - File data blocks.

File System Structures In Memory

- Mount table.
- Directory-structure cache.
- System wide open file table.
- Per-process open file table.

File System Structures



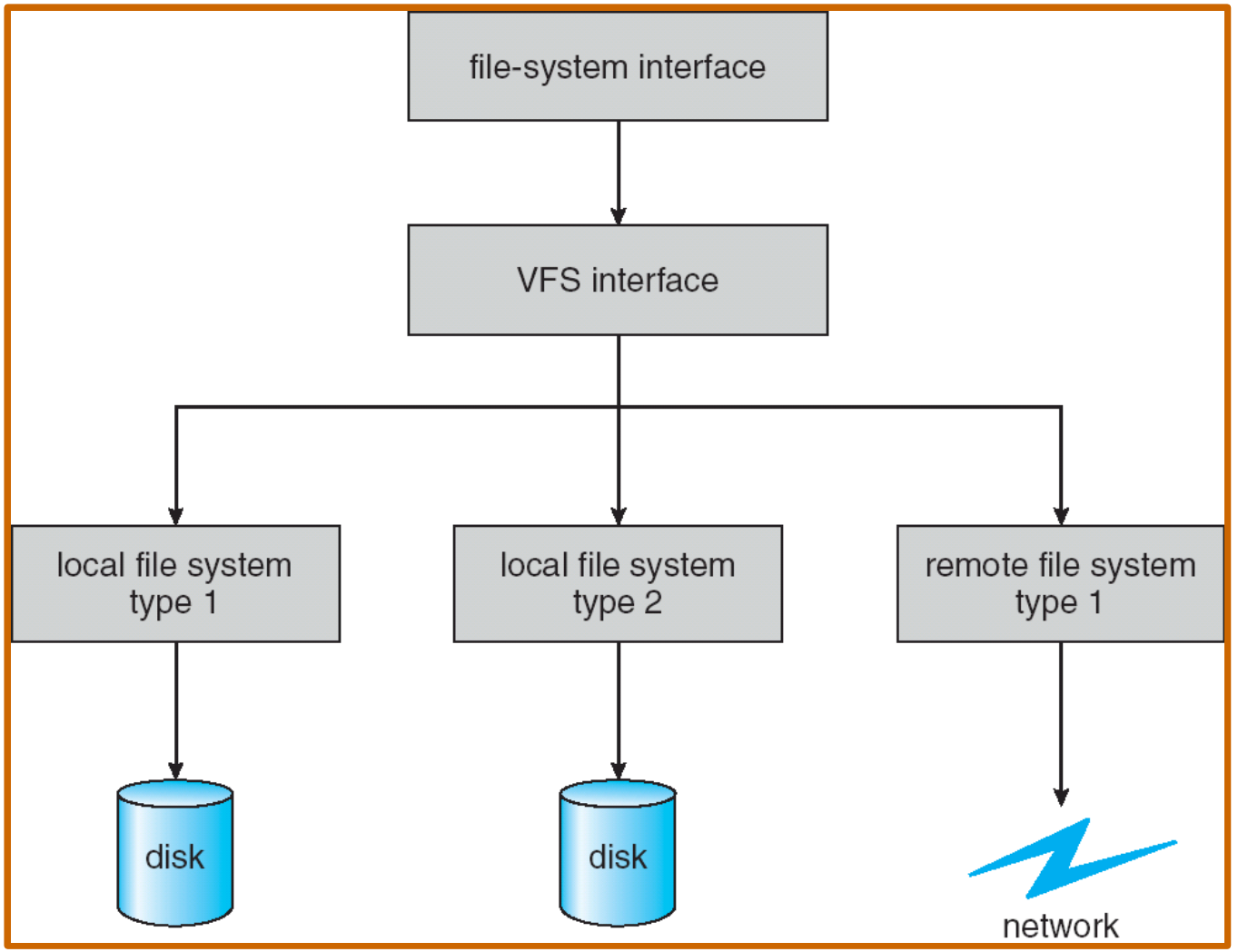
Virtual File Systems

- There are many different file systems.
  - Windows machines support FAT, NTFS, WinFS (coming soon...)
  - Linux Supports dozens of file systems.
- OS would like to present a single file interface to users.
- OS would like to present a single file interface to *itself*.
- Solution is VFS.

VFS

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.
- Makes it possible to mix and match different file systems transparently.

VFS Picture


file-system interface
VFS interface
local file system type 1
local file system type 2
remote file system type 1
disk
disk
network

Directory Implementation

- A directory can be thought of as a special file that only contains references to other files.
  - How should such a file be organized?

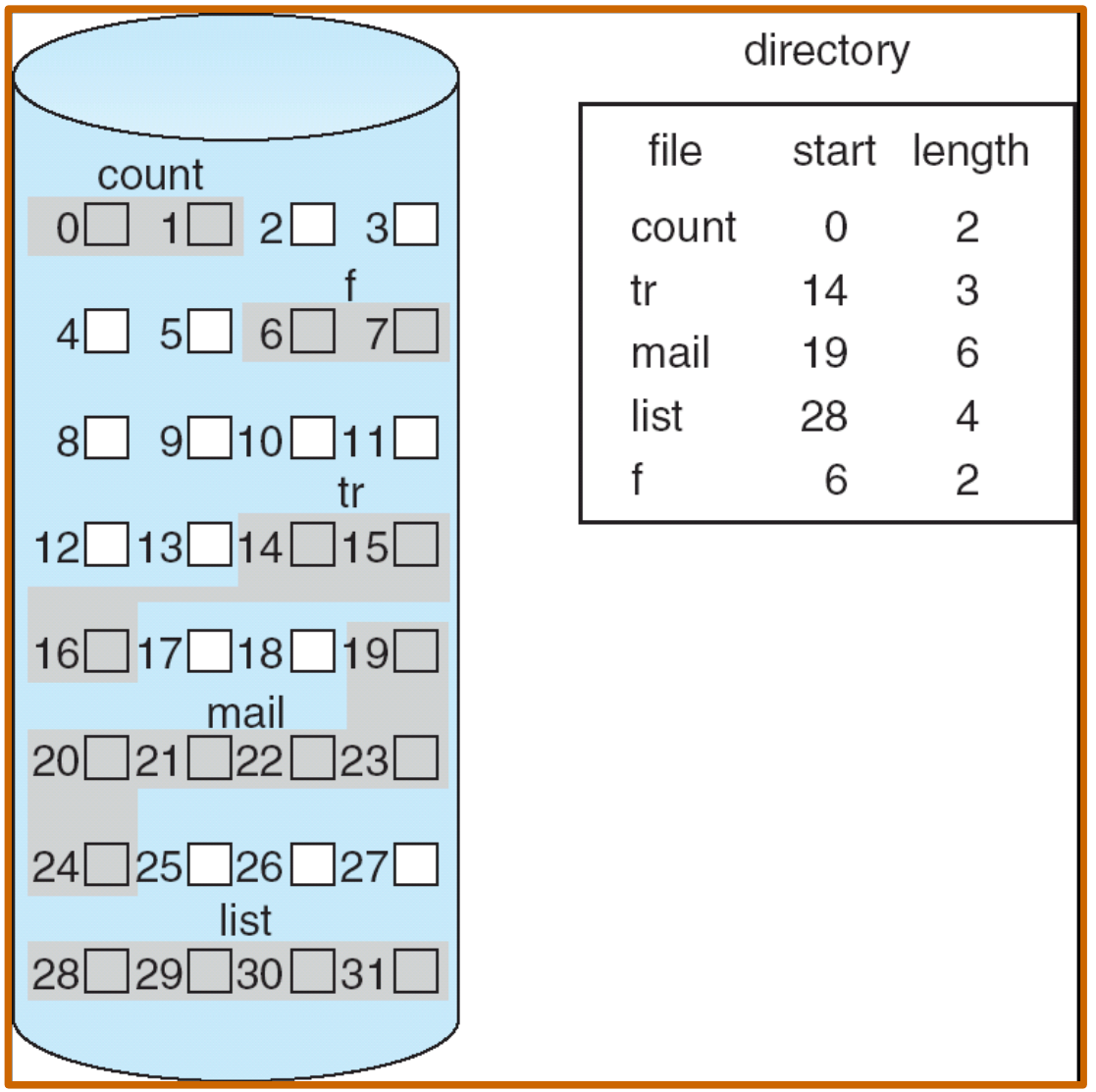
Directory Implementation

- A directory can be thought of as a special file that only contains references to other files.
  - How should such a file be organized?
- Linear List:
  - Simple to program.
  - Some operations will be slow.
  - Might help to keep it sorted.
- Hash table:
  - Fast search.
  - Need to handle collissions.
  - Sizing and resizing can be challenges.

Allocation Methods: Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required for access.
- Random access.
- Good performance.
- Problems?

Contiguous Allocation Picture



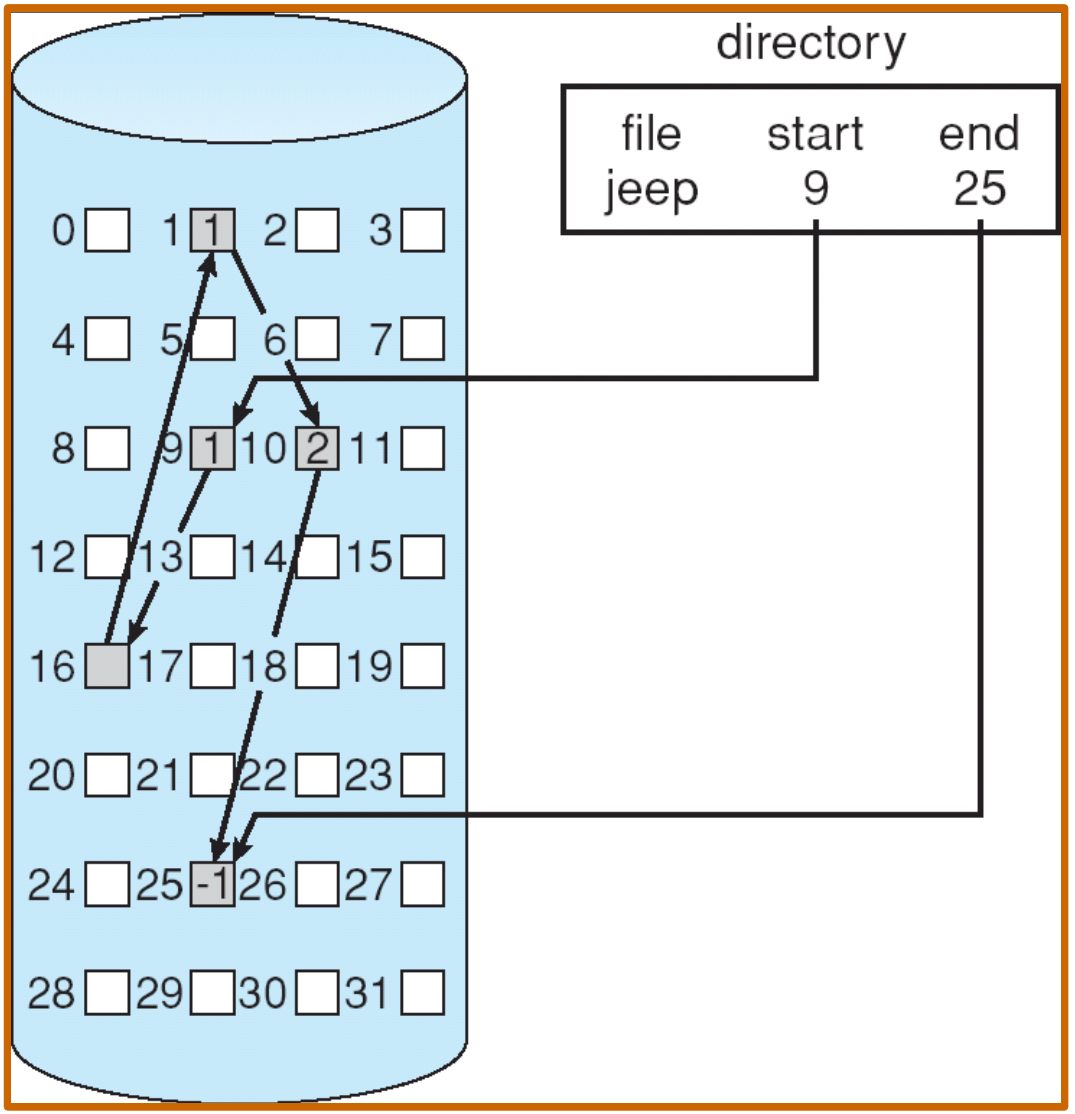
Extents

- Some newer file systems (I.e. Veritas File System) use a modified contiguous allocation scheme.
- Allocate disk blocks in **extents**.
- An extent is a contiguous range of blocks.
  - Single extent is initially allocated.
  - A file consists of one or more linked extents.

Linked Allocation

- Each file is a linked list of disk blocks.
- Blocks may be scattered anywhere on the disk.
  - Simple.
  - No external fragmentation.

Linked Allocation



- Drawbacks to Linked Allocation?

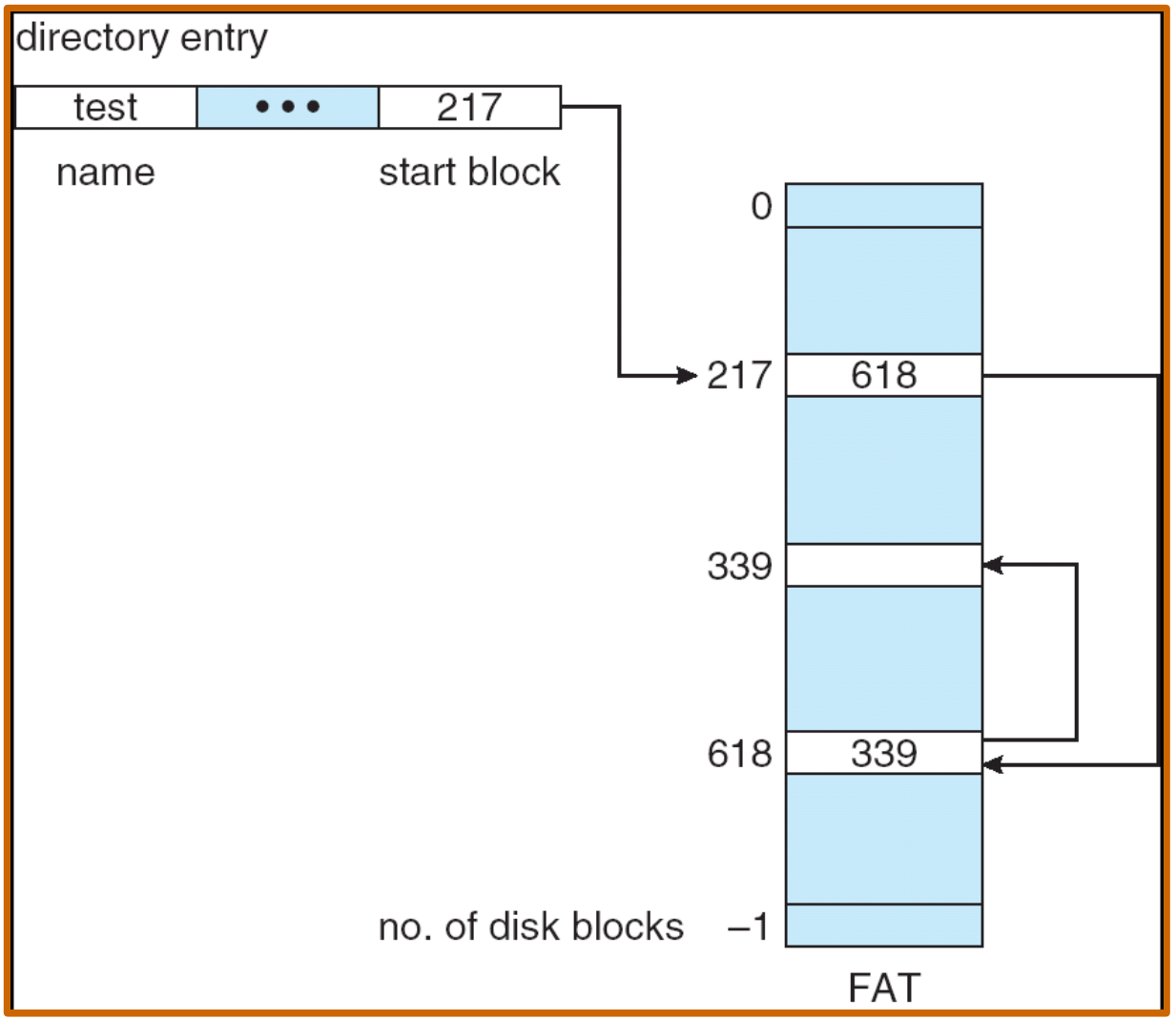
Linked Allocation Drawbacks

- No random access.
- Significant space wasted on links.
- Reliability.

File Allocation Tables

- Table at the beginning of a volume with one entry for every block.
- Each entry may contain a link to the next block in a file.
- A variation on linked allocation that enables random access.

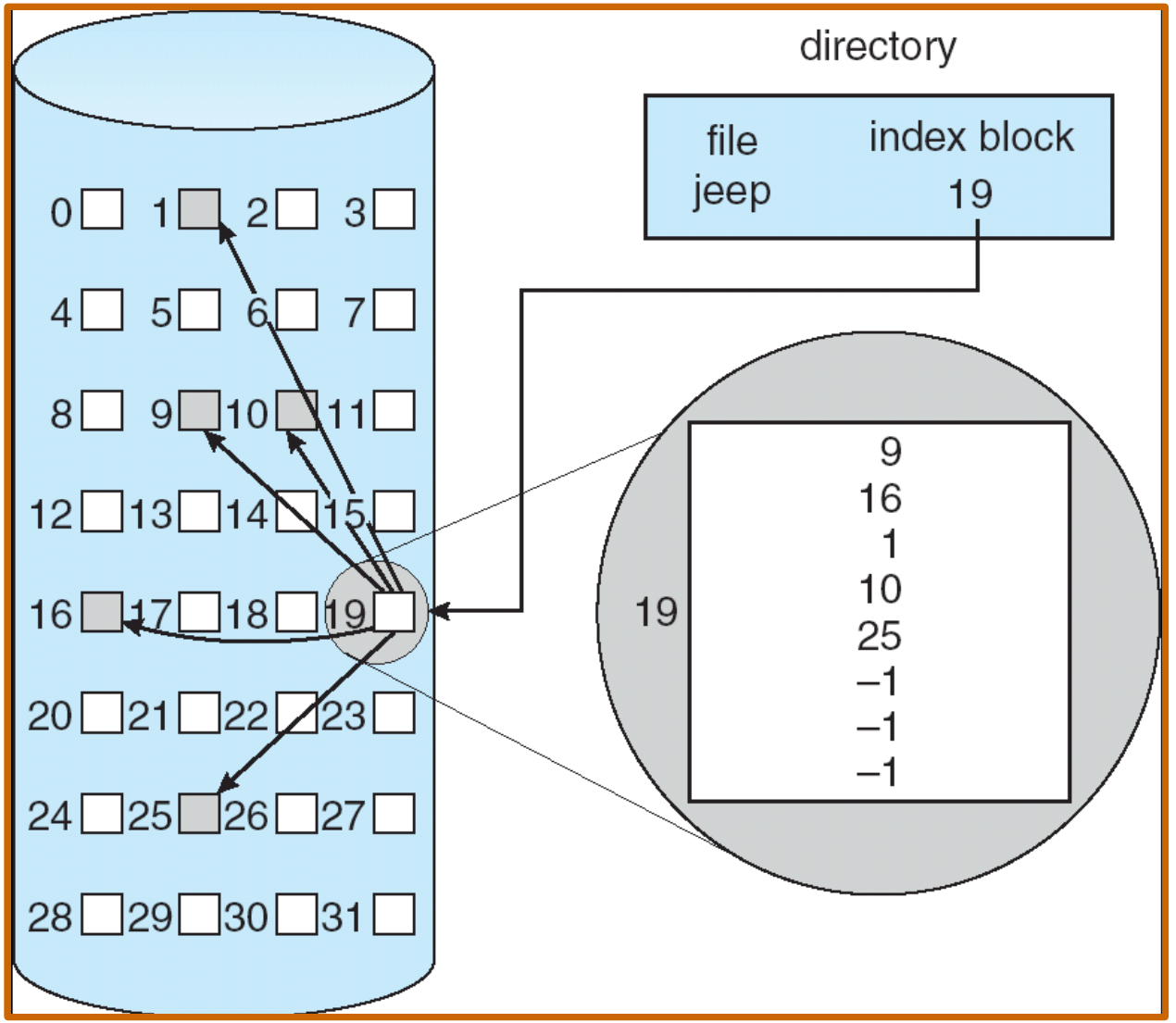
FAT Example


directory entry
test
• • •
217
name
start block
0
217
618
339
618
339
no. of disk blocks
–1
FAT

Indexed Allocation

- Each file has an index block that contains pointers to data blocks.
  - Good random access.
  - No external fragmentation.
- Similar to the page table idea.

Indexed Allocation

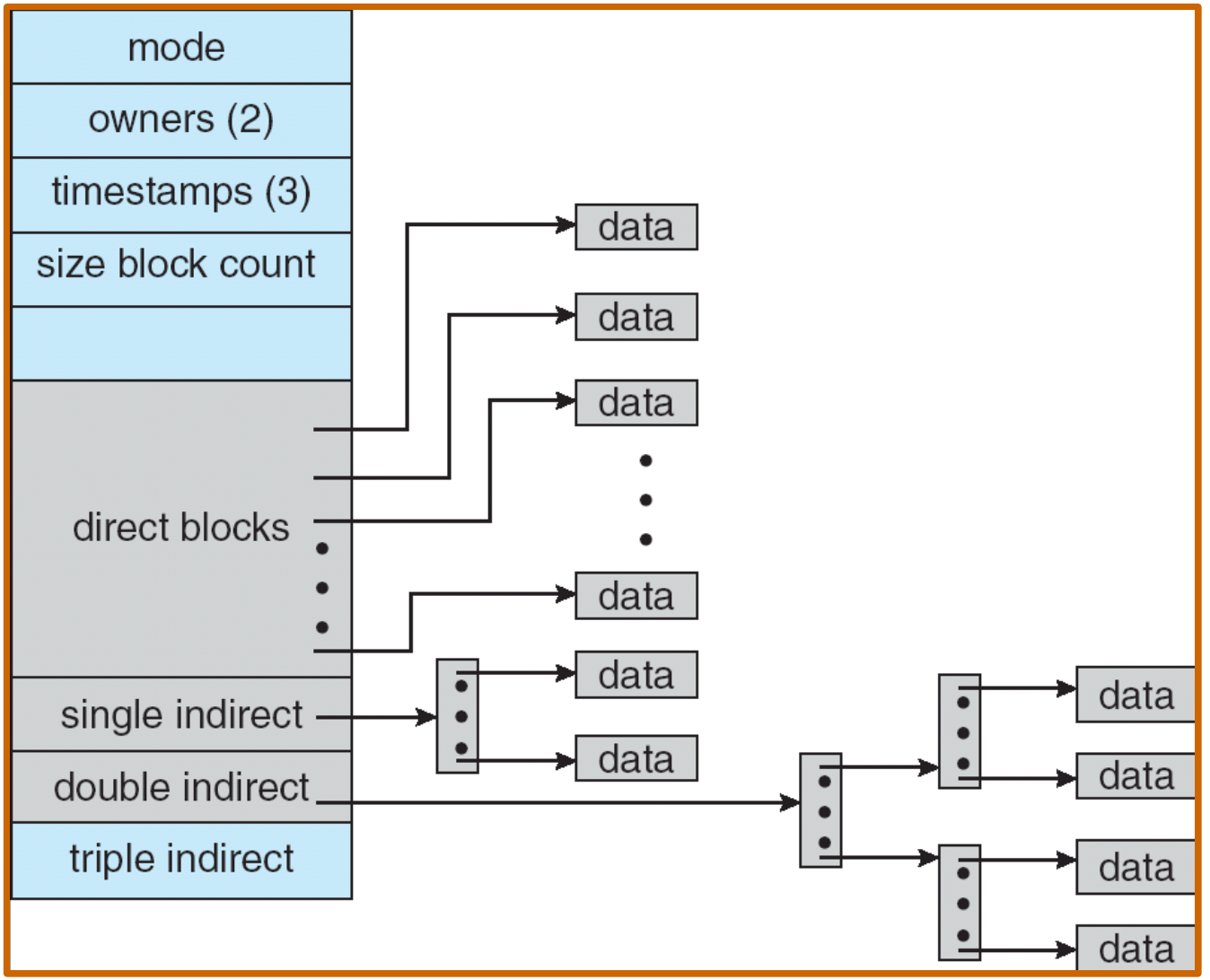


- Drawbacks?

Indexed Allocation

- Drawbacks:
  - Wasted space for small files. (Index block is mostly empty.)
  - Limited file size.
- Solutions to the second problem:
  - Link multiple index blocks.
  - Multi-level index blocks.
  - Combined single level and multi-level scheme.

Unix File System (UFS) Index Blocks



Free Space Management

- Bit Vector
  - Requires extra space:

    block size = 2^{12} bytes

    disk size = 2^{30} bytes (1 gigabyte)

    $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
  - Easy to get contiguous files.
- Linked list (free list).
  - Cannot get contiguous space easily.
  - No waste of space.
- Grouping...
- Counting...

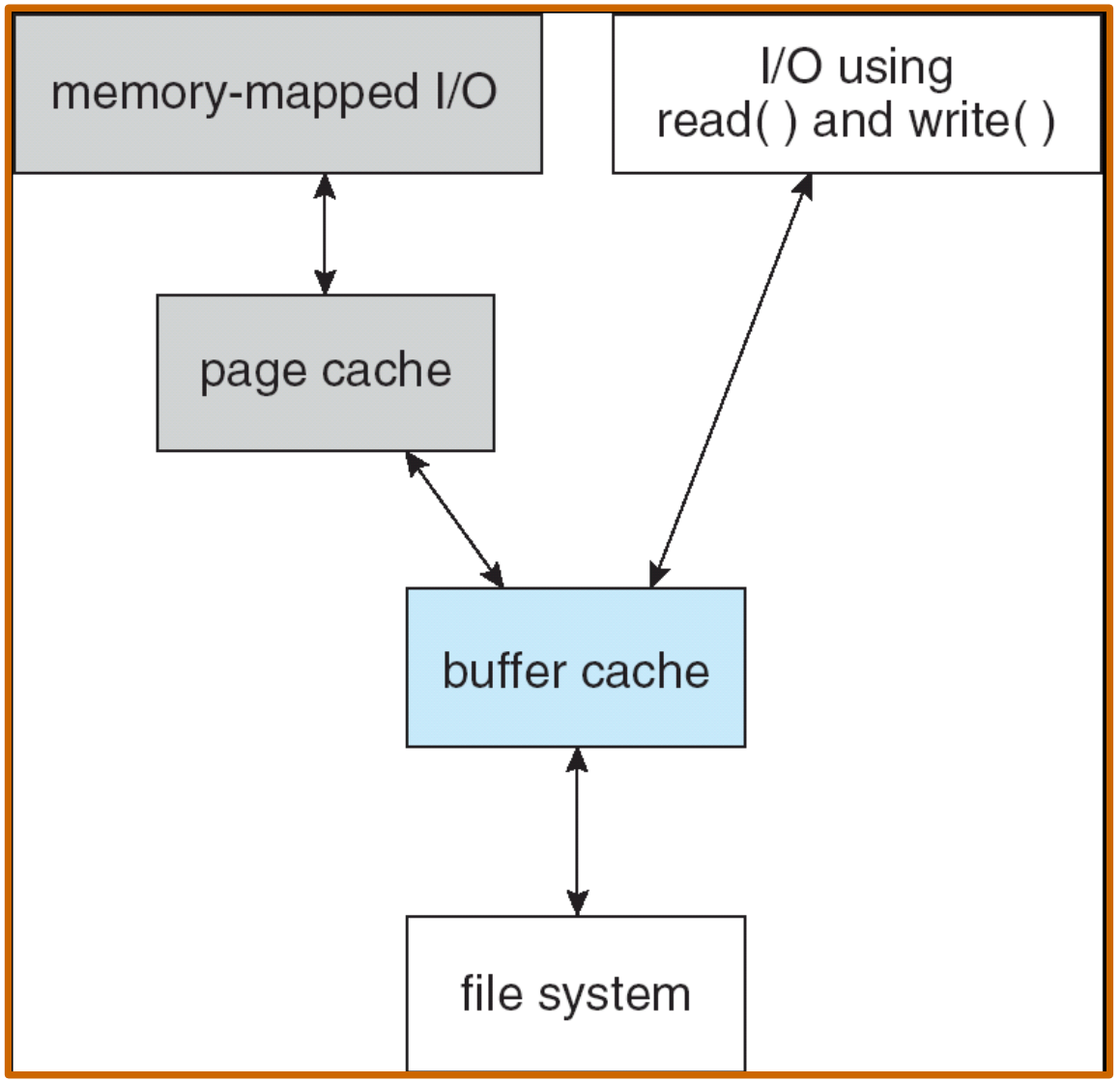
Performance

- UFS pre-allocates inodes (index blocks) and spreads them across the disk.
  - Costs disk space.
  - Helps keep data blocks near inode block.
- Disk cache – separate section of main memory for frequently used blocks.

Disk Cache vs. Page Cache

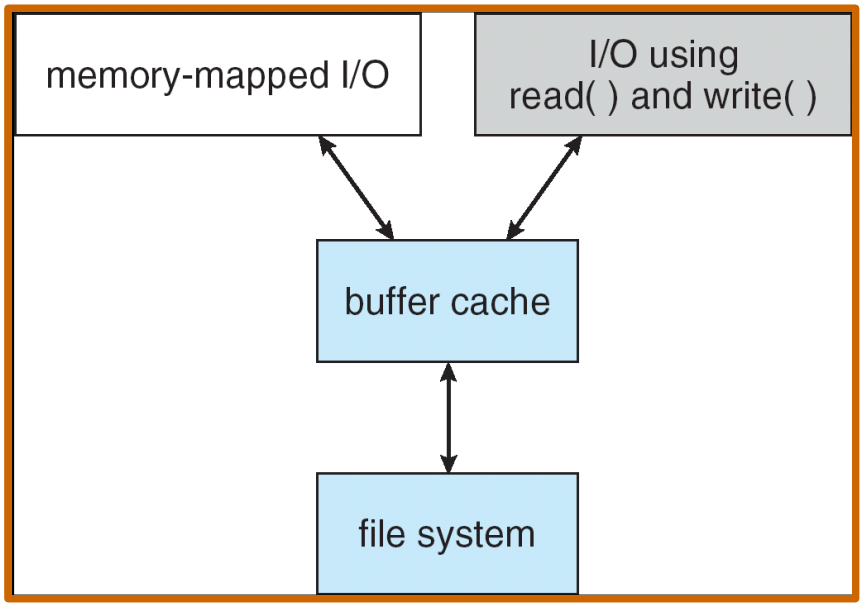
- A **page cache** caches pages rather than disk blocks using virtual memory techniques.
- Memory-mapped I/O uses a page cache.
- Routine I/O through the file system uses the buffer (disk) cache.
- Aside: Synchronous vs. Asynchronous access.
- This leads to the following figure.

Page Cache and Buffer Cache



- Any problems with this?

Unified Buffer Cache



- Who should get priority in the cache, file system data or process memory?
- LRU or something else?
  - Free behind, read ahead may be appropriate for sequential access.

Recovery – Consistency Checking

- Because of computer crashes or device errors file systems can contain inconsistent information.
  - To try to avoid this, modifications to the file system structure are usually performed synchronously.
- Many file system include a mechanism to do consistency checks:
  - Unix: fsck
  - MS-Dos: chkdsk
- These are run periodically, or when a problem is suspected.

Log Structured File Systems

- Log structured (or journaling) file systems record each update to the file system as a transaction.
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log.
  - However, the file system may not yet be updated.
- The transactions in the log are asynchronously written to the file system.
  - Results in a performance gain. (WHY?)
  - When the file system is modified, the transaction is removed from the log.

Acknowledgments

- Portions of these slides are taken from Power Point presentations made available along with:
  - Silberschatz, Galvin, and Gagne. Operating System Concepts, Seventh Edition.
- Original versions of those presentations can be found at:
  - http://os-book.com/