CSc 360
Operating Systems
File Systems

Wenjun Yang

Fall 2025

Review

- Operating systems
  - process management
    - how to share CPU: jobs, processes, threads
    - scheduling, communication, synchronization
- What's next?
  - **file systems (this and next week)**
  - memory management
  - I/O systems

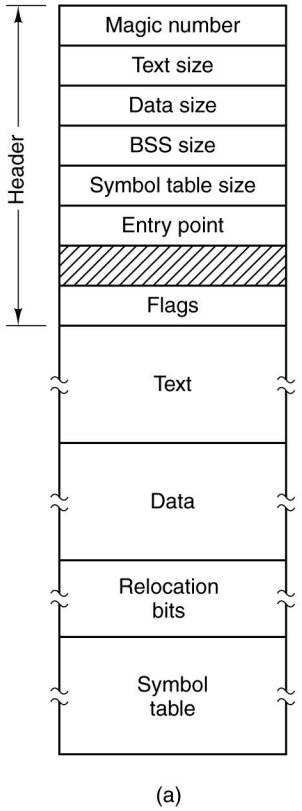
File concepts

- File: logical storage unit for information
  - normally on disk and tape: nonvolatile
- File type: data (text, binary, …), program
- File attributes: meta information
  - name, size, time, owner, protection, location, …
  - kept in directory (a special file)
- File operations
  - create, write, read, seek, append, truncate, etc

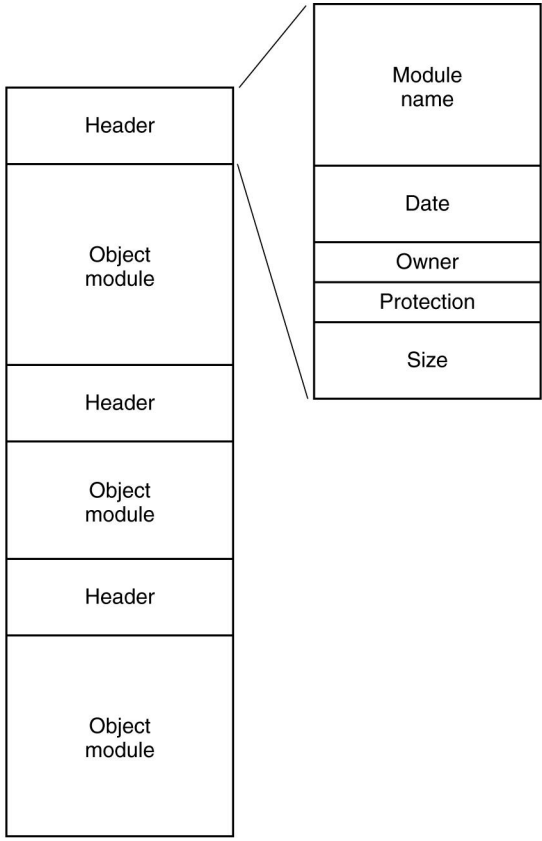
. and ..? *.hidden*?

File structures

- How information organized within a file
  - unstructured, or
  - structured w/ records
- Example
  - binary executable
  - binary archive
    - modules



(a)



(b)

$ file p1.pdf
p1.pdf: PDF document, version 1.4

$ head p1.pdf
%PDF-1.4

File access

- Sequential access
  - variable length records, i.e., r_1, r_2, …, r_n
  - to access record r_i, go through r_1 to r_{i-1} first
- Direct access
  - fixed length records of size s
  - i-th record: offset (i-1)*s
- Indexed access
  - access index first

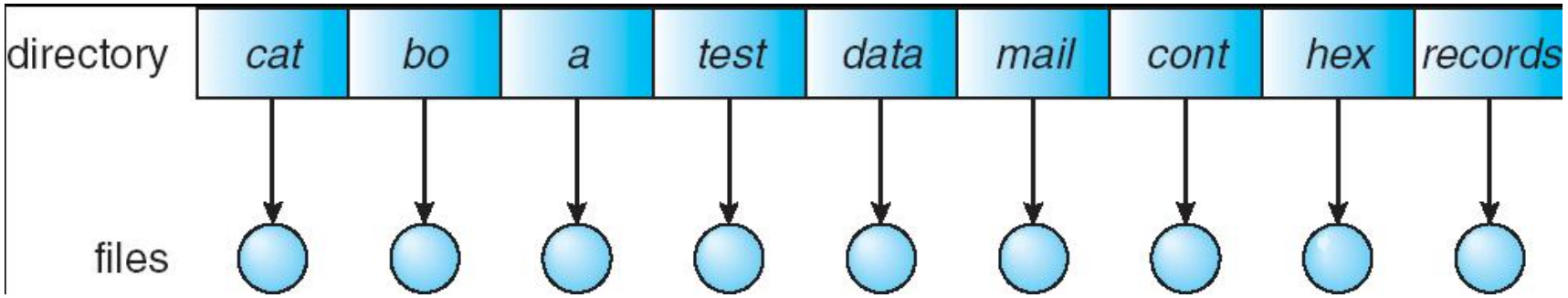
Directory concepts

- A special file
  - a collection of pointers to meta information of files
  - normally structured
  - and allow direct access
- Directory operations
  - “file” operations on directory records
- File systems on disk
  - partitions, file systems (FDT/FAT), files

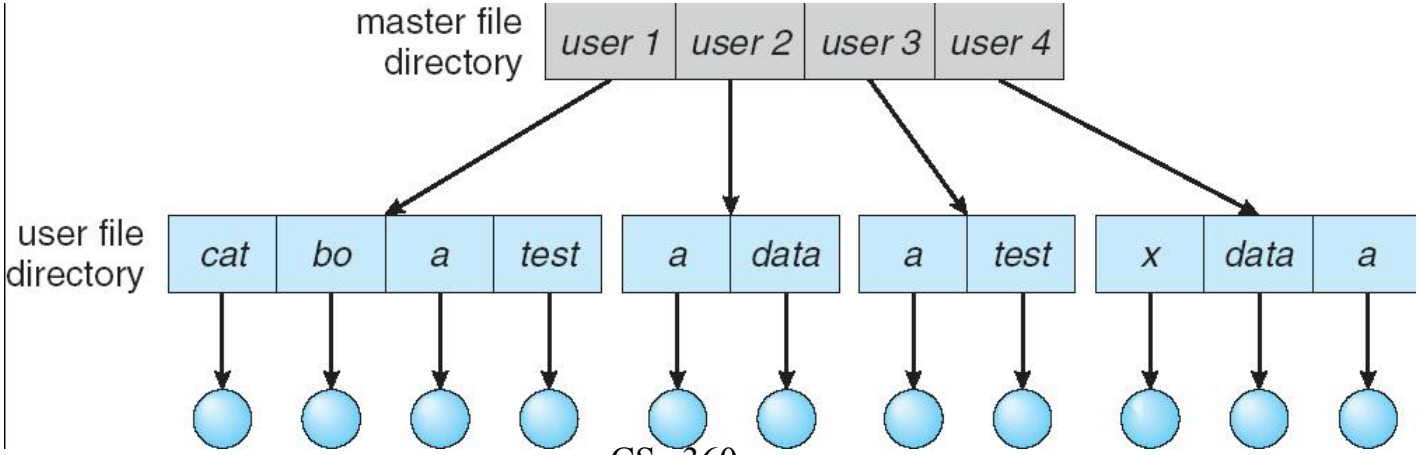
partition vs volume?

Directory structures

- Single-level directory

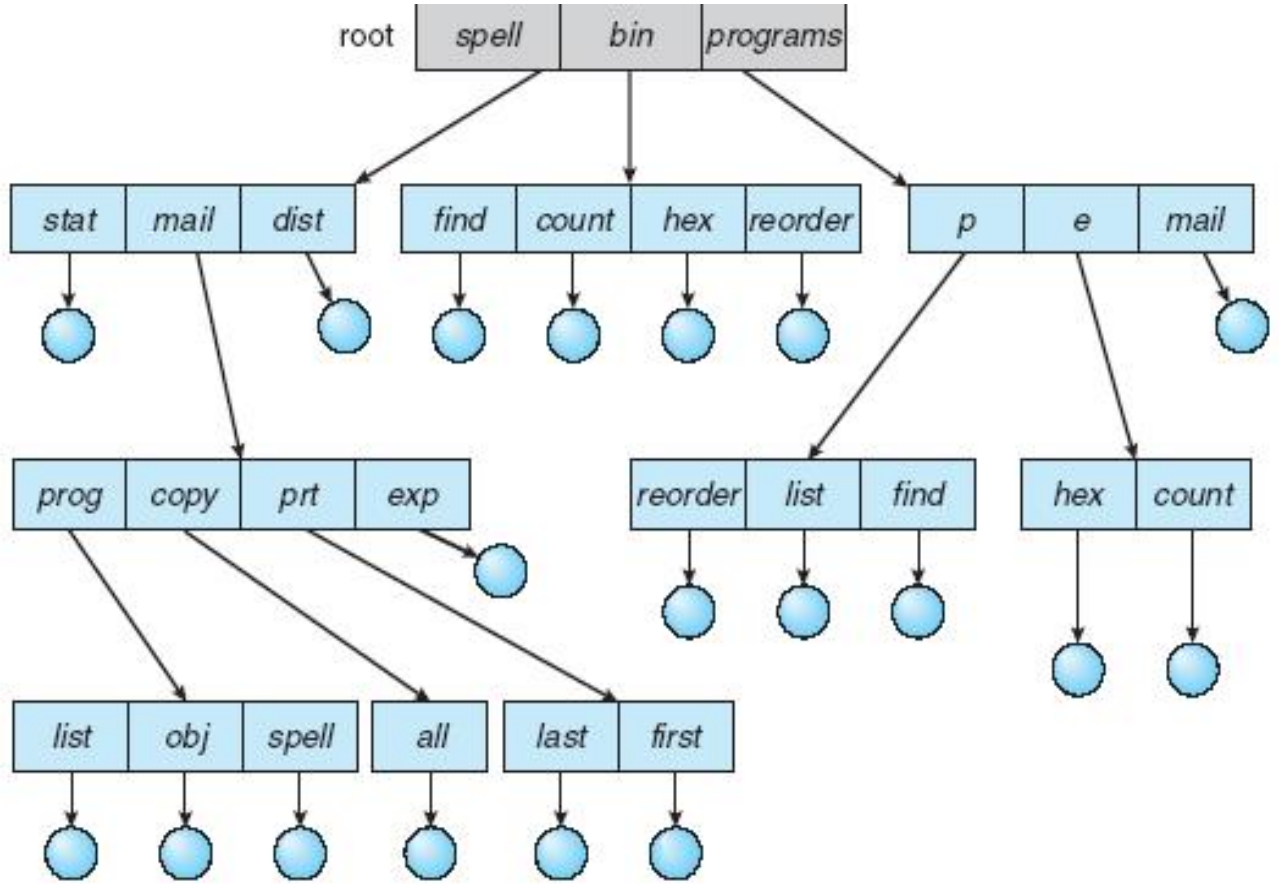


- Two-level directory



Directory structures: tree

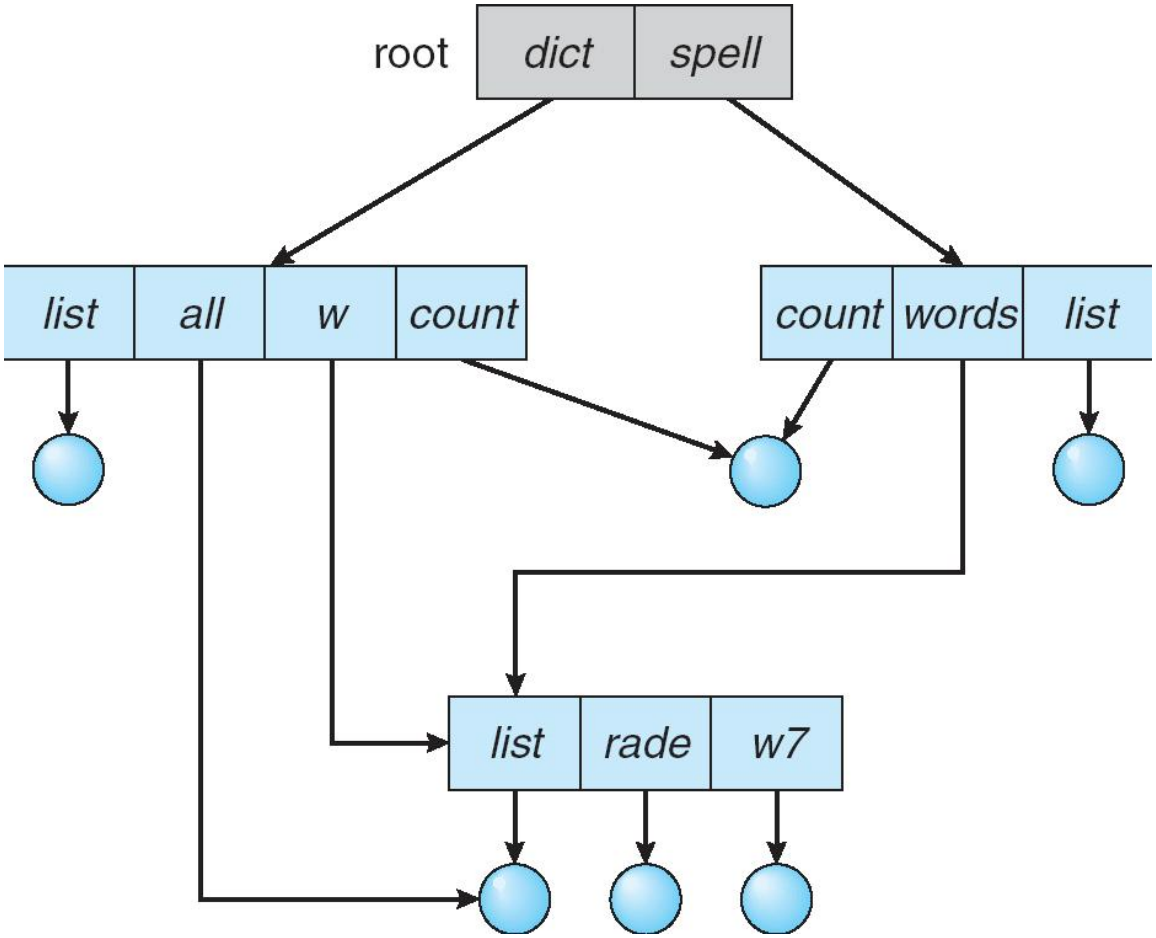
- Path
  - absolute
  - relative



cd in p1?

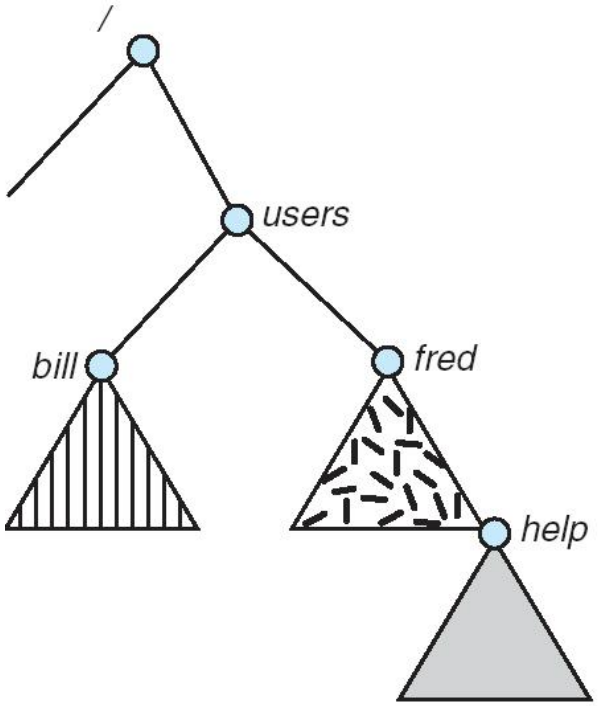
Directory structures: graph

- Links
  - hard links
  - symbolic links
- Link operations
  - resolve links
  - delete links/files
- Acyclic-graph
  - loop-free

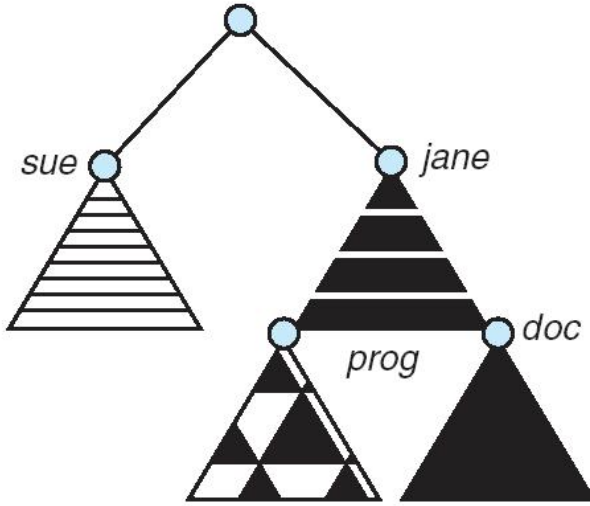


File system operations

- Mount file systems
  - mount point
    - directory
    - directory with files?
  - multi-mount



(a)



(b)

same home directory in different teaching labs?

Share and protection

- File attributes
  - owner: user ID, group ID
- Type of access
  - read, write, execute, etc
- Share over the network
  - host ID
  - NFS: network file systems
    - remote procedure calls

Failure mode

- Local file systems
  - hardware failure
  - corrupted information
- remote file systems
  - network failure
- Fault-tolerant systems
  - keep state at both sides, or
  - stateless but less secure

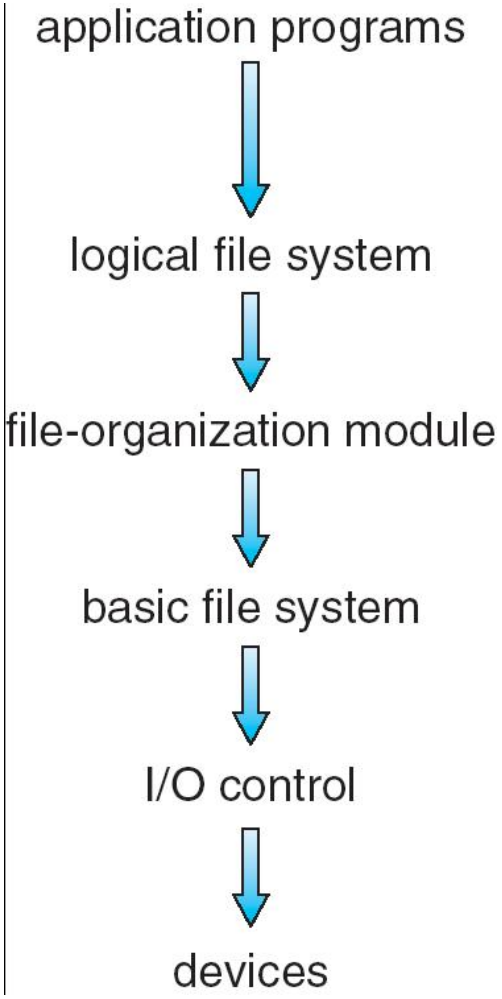
Consistency control

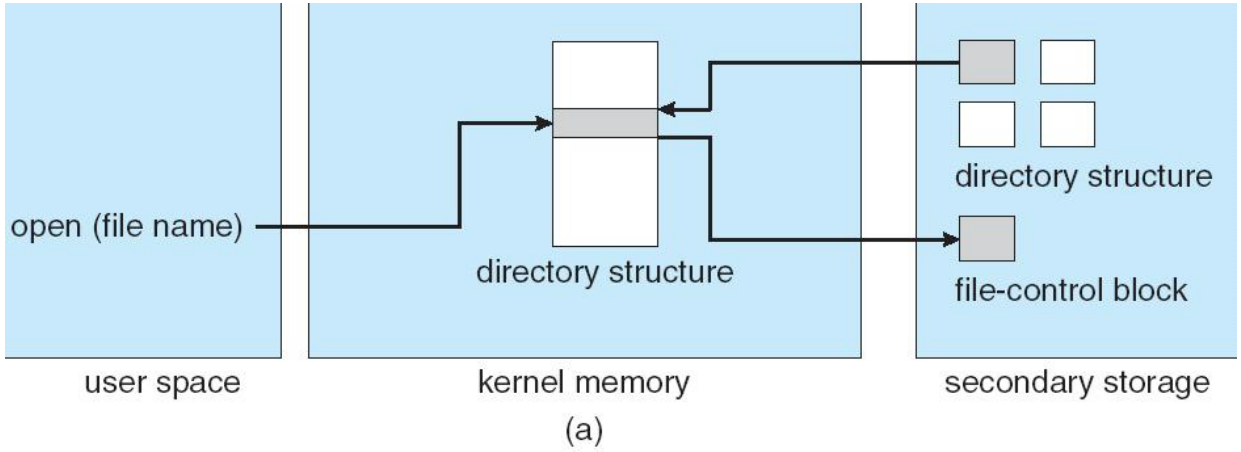
- Multiple users access a file simultaneously
  - lock!
    - entire file, or
    - certain records
  - complexity and (network) overhead
- Consistency semantics
  - Unix semantics: write immediately visible
  - session (open()-close()) semantics: e.g., AFS

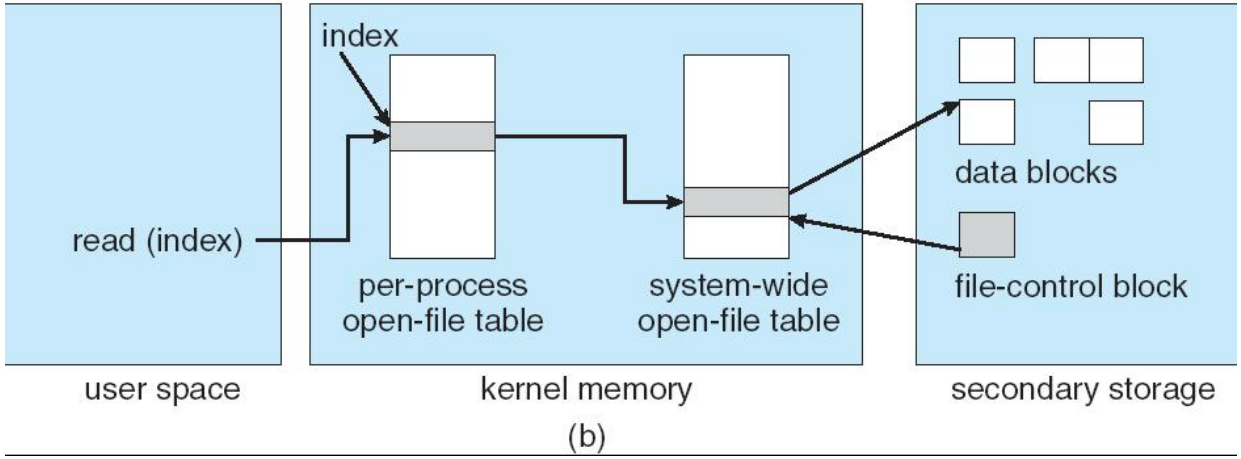
File system organization

- Sector: e.g., 512 bytes
- Block
  - data blocks
  - control blocks
    - boot, partition, (root) directory, file control blocks
- Access blocks
  - logical block address
  - physical block address: disk geometry

File system structures

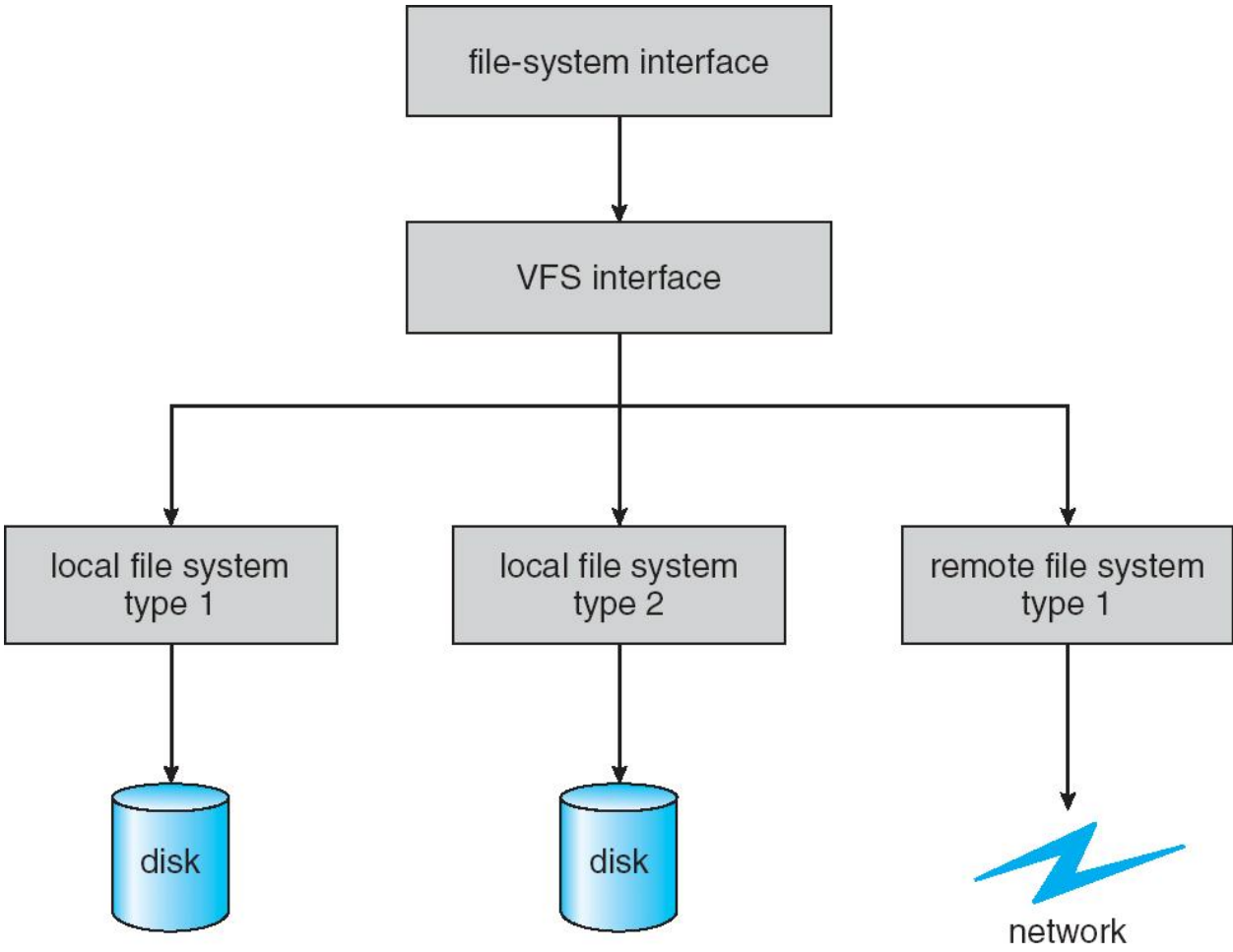






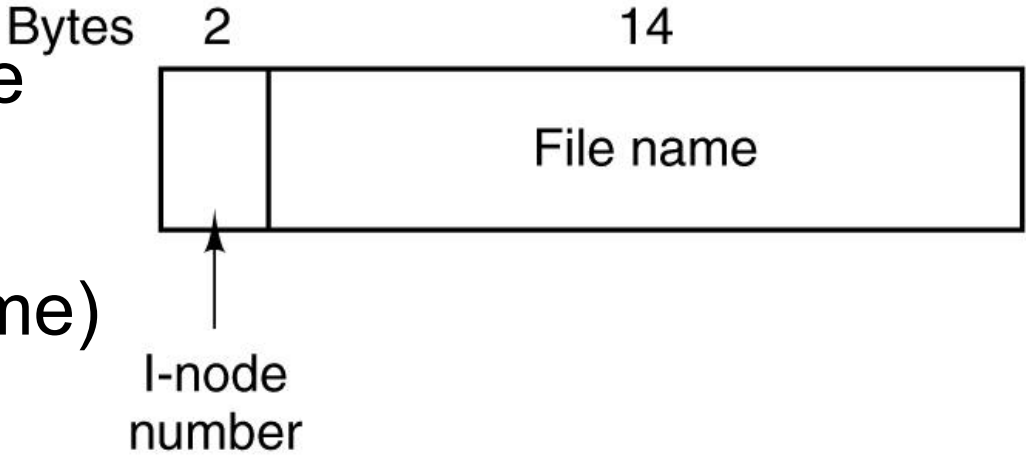
Virtual file systems

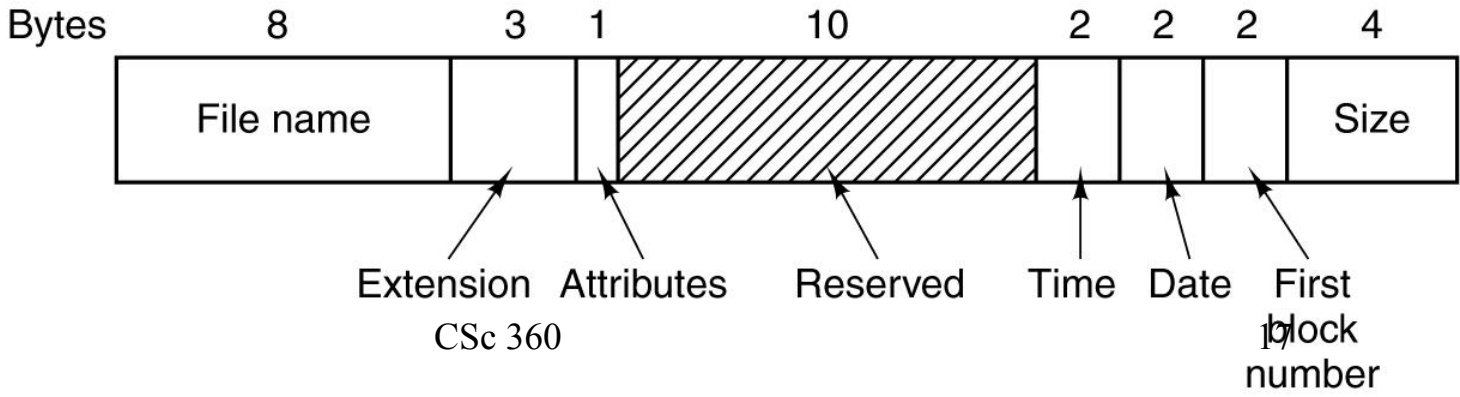
- OO design
  - same API to different file systems
  - e.g., open(), read(), write(), etc



Implementing directory

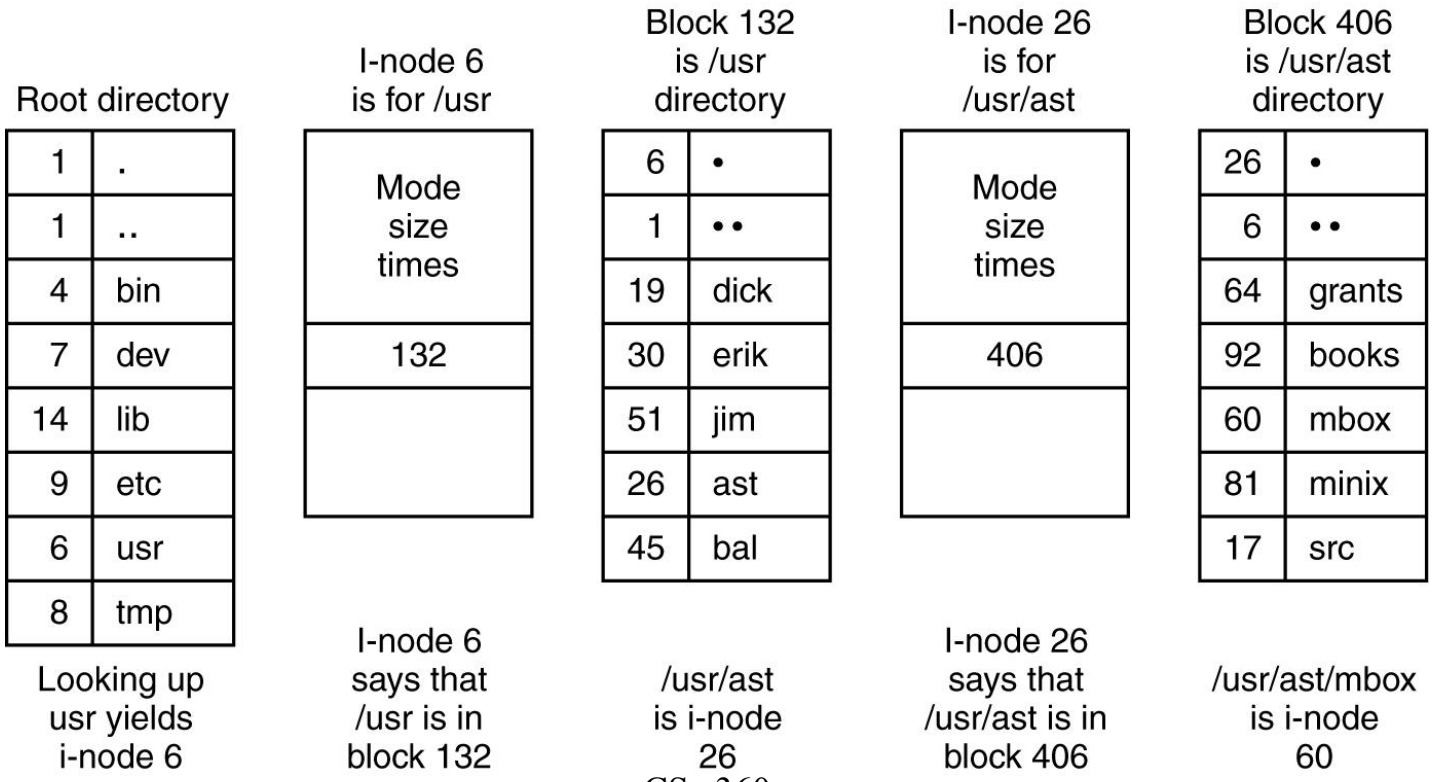
- Directory table
  - linear list or hash table
- Directory entry
  - e.g., Unix (i-node, name)
  - e.g., DOS





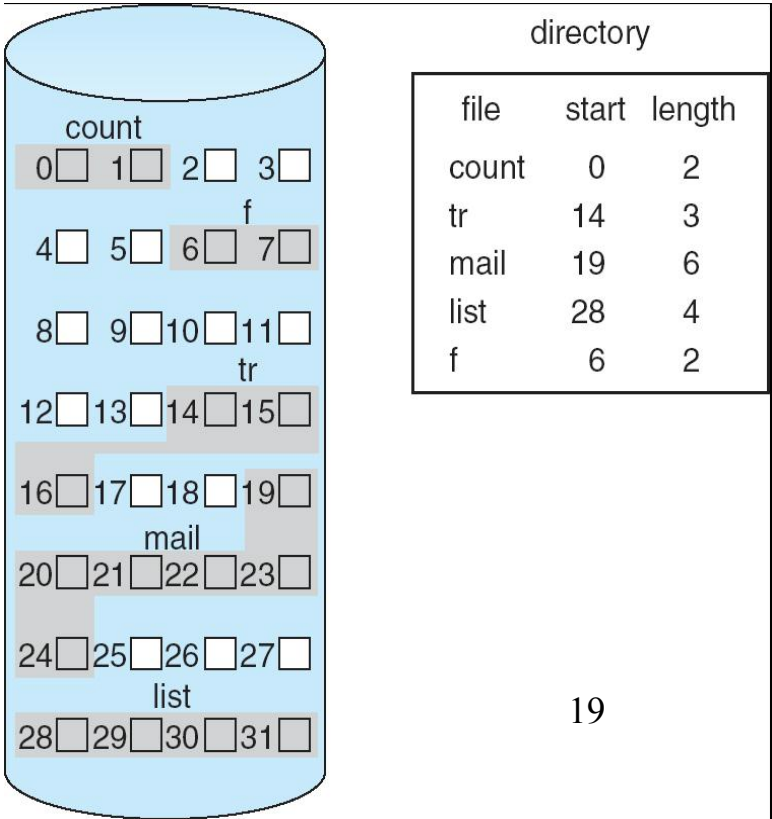
Traversing directory hierarchy

- Unix: /usr/ast/mbox



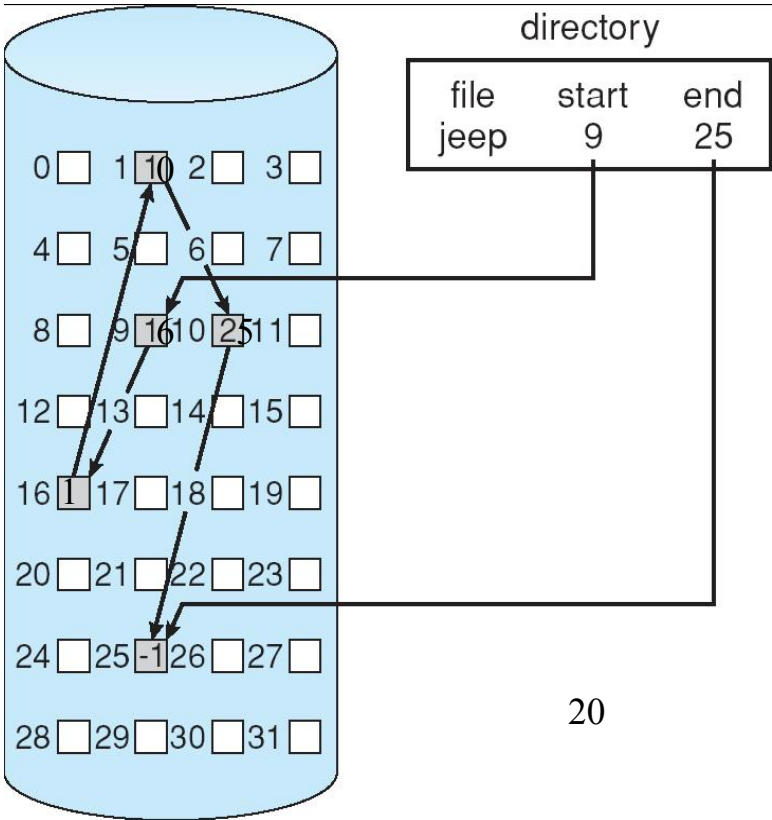
Allocating blocks

- Contiguous allocation
  - address of the first block, length
  - random access
  - difficult to expand
  - fragments
- Modified contiguous allocation
  - e.g., extent-based allocation



Linked allocation

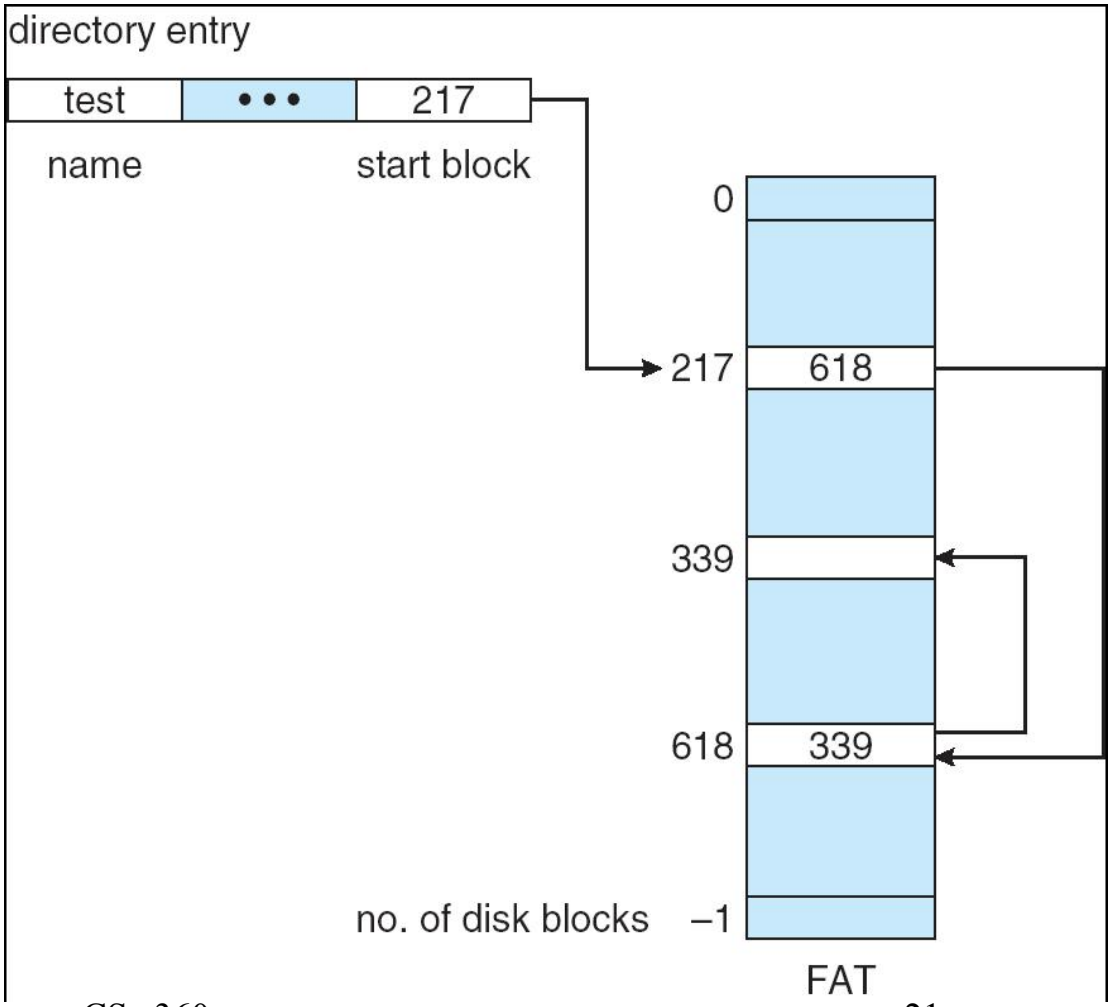
- Linked list
  - address of the first block
  - each block has a next pointer
    - next == -1, end of block list
  - easy to expand
  - no direct random access
- Modified linked list
  - e.g., cluster-based allocation



* 9 -> 16 -> 1 -> 10 -> 25 -> -1 (EOF)

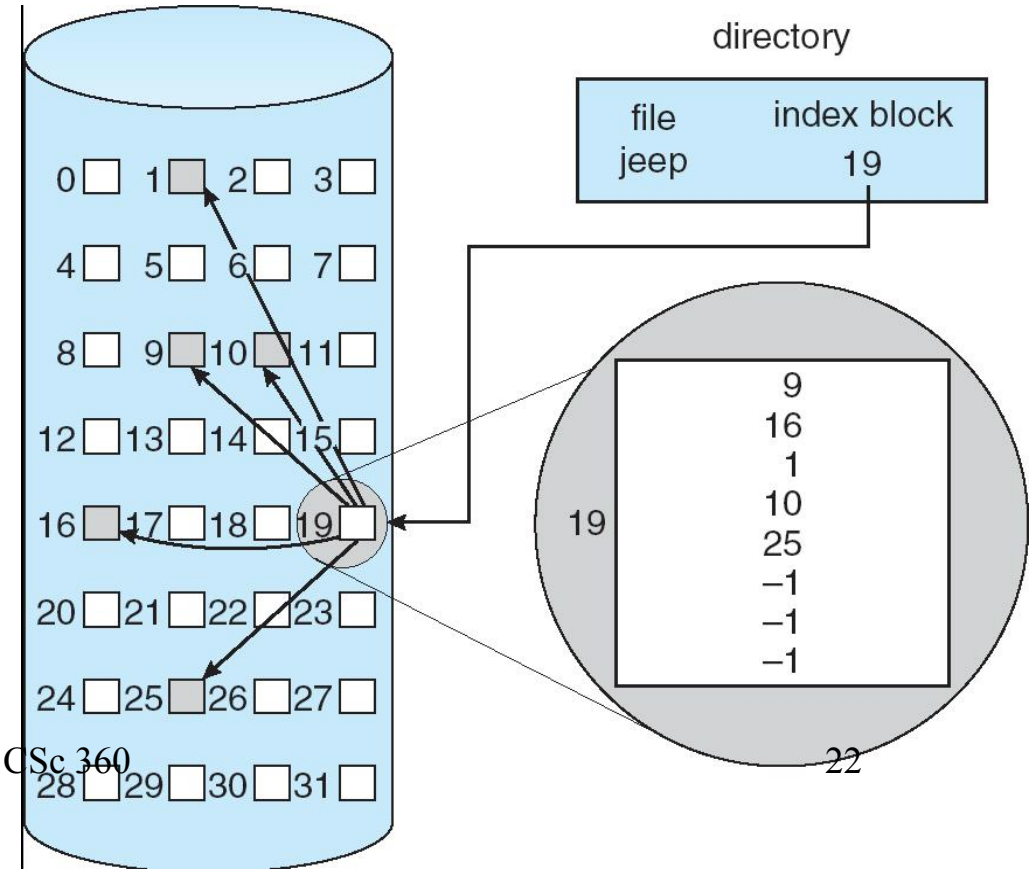
File allocation table

- FDT
  - first block address
- FAT
  - linked list of addresses
  - can be cached
    - random access



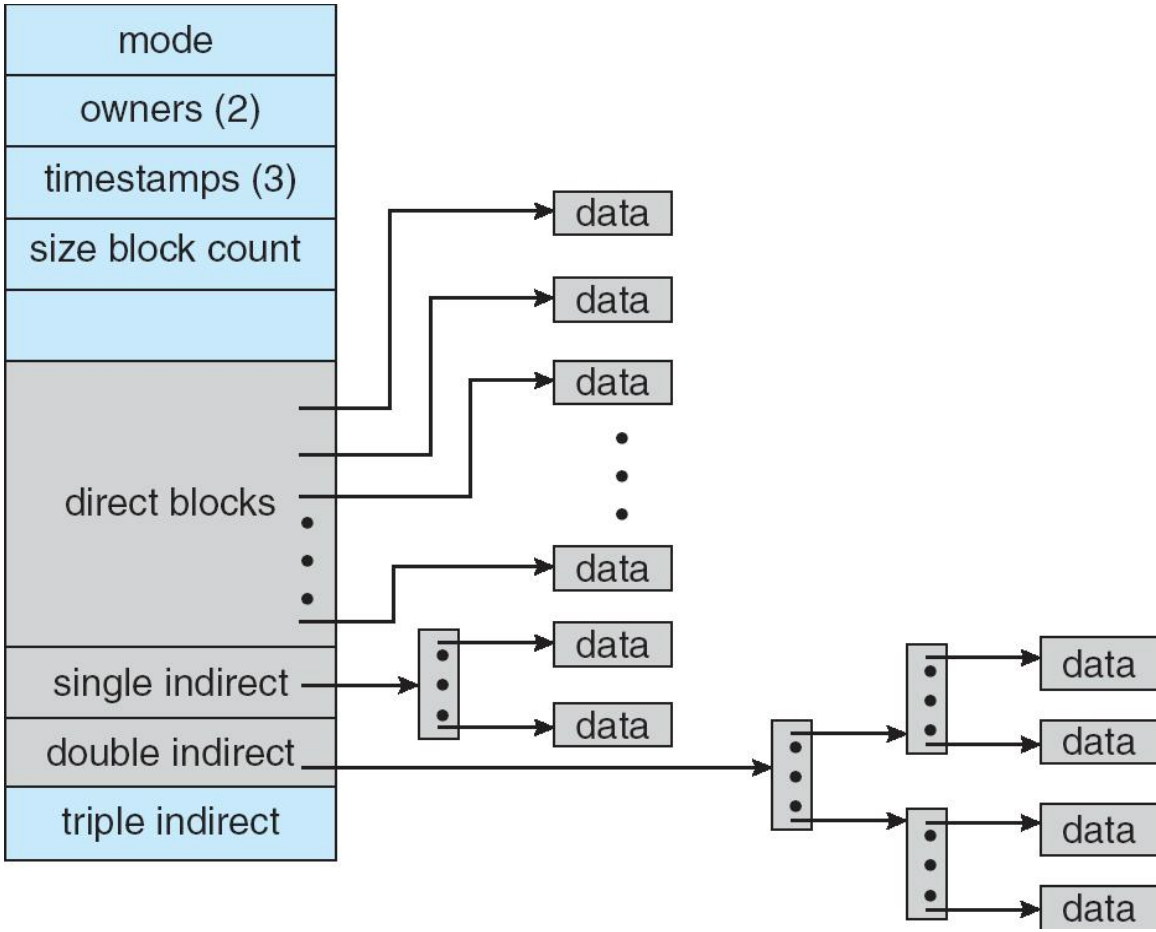
Indexed allocation

- Index block
  - a list of pointers
  - random access
  - easy to expand
  - limited by the size of index block



Indirect index

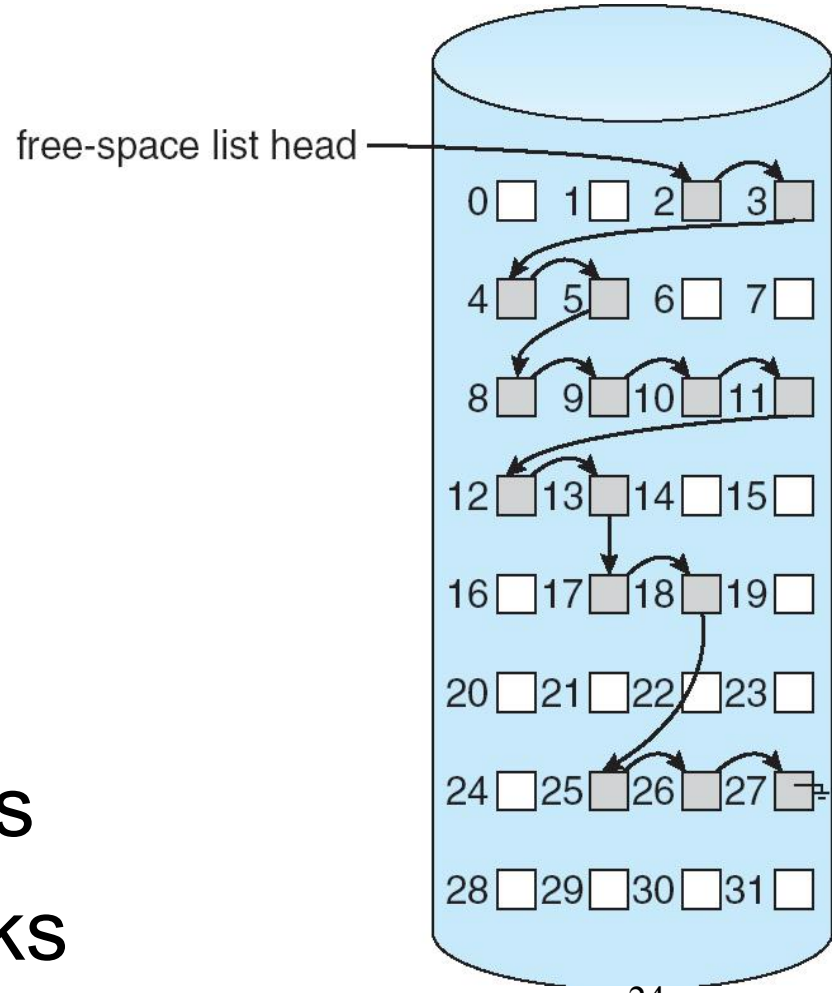
- E.g., Unix i-node
  - 12 direct index
  - 1 single indirect
  - 1 double indirect
  - 1 triple indirect
  - 4KB block



* every cs problem can be solved by another layer of indirection ;-)

Free space management

- Keep track of free blocks
- Bitmap
  - 1 bit indicates whether a block is free or not
- Linked-list
  - “a big free file”
- Grouping: list of free blocks
- Counting: contiguous blocks



* How does the FAT system do this?

Efficiency and performance

- Efficiency: tradeoff
  - e.g., FDT/FAT/block size, # of blocks/entries
  - static vs dynamic data structure
- Performance: further tradeoff
  - buffer vs page cache
    - unified buffer cache
  - free-behind, read-ahead: sequential access
  - RAM disk

Consistency check

- Inconsistency
  - between files and their meta-information
    - due to hardware/software failure
- Consistency check
  - e.g., fsck: file systems consistency check
    - superblock consistency
      - wrong counters
    - inode consistency
      - unreferenced inodes, missing free blocks, bad *free* blocks

* try something similar but for FAT in P3 too

Backup and restore

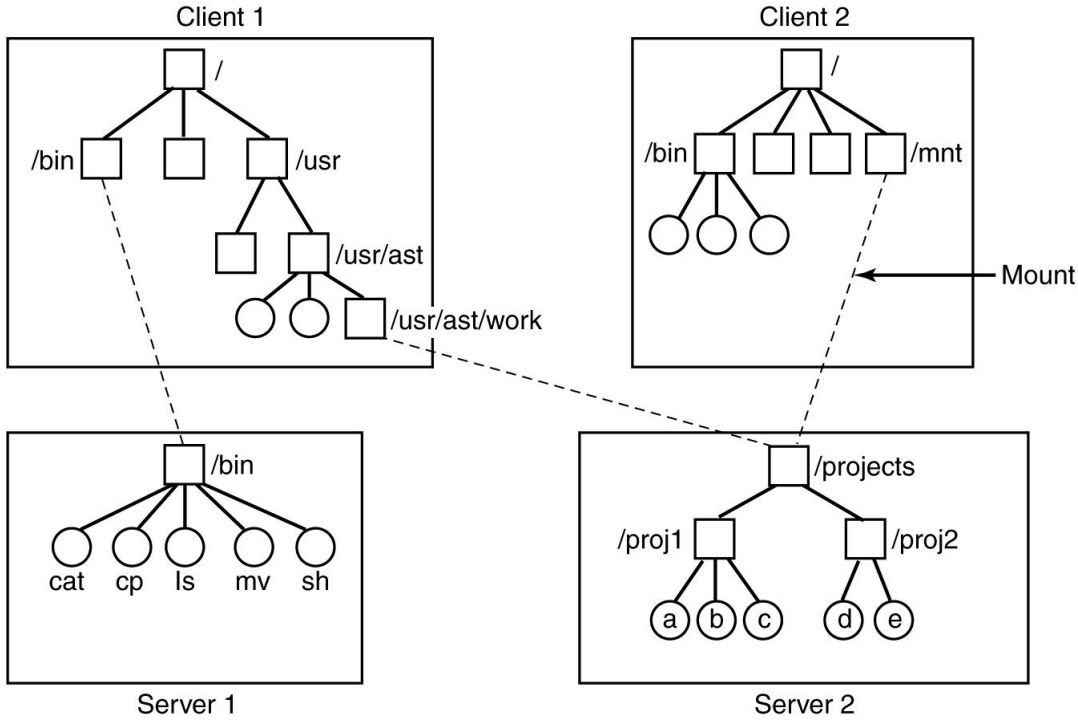
- Backup to another medium
  - magnetic, optical, etc
- Full backup
- Incremental backup
  - last backup time
  - last modify time
- Backup cycle
  - full backup, {incremental backup}*

Log-structured file system

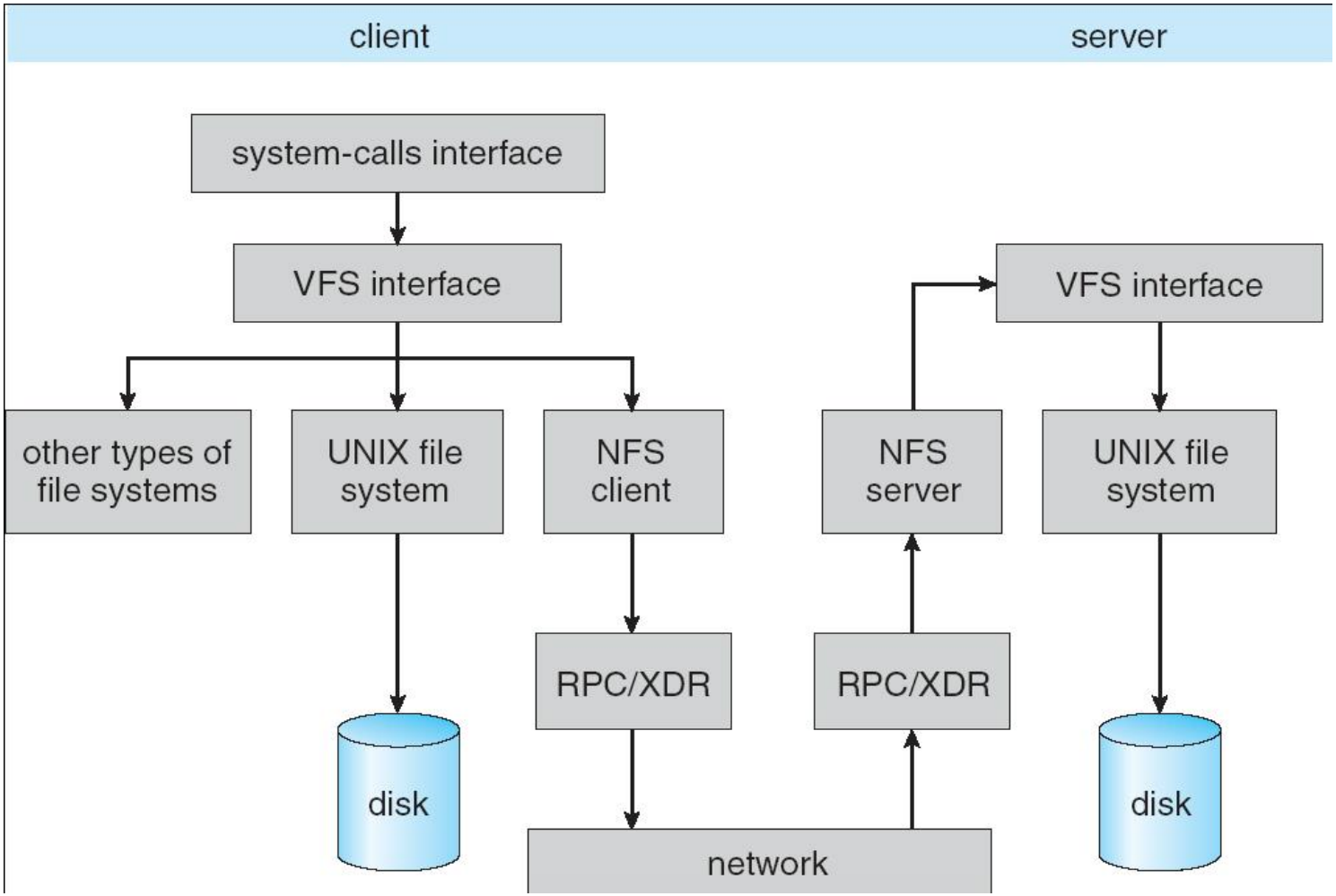
- Transactions: updates to file system
  - considered committed when written to a log
    - file system may not be updated yet
  - asynchronously update file system
    - performance improvement as well
- After system crashes
  - transactions still to be completed
    - file system remains consistency

SunOS NFS

- Access remote file system
- Initially on UDP
- With RPC/XDR
- Transparency
- Mount protocol
  - server: exportfs
    - access control
  - client: mount server:dir dir



NFS architecture



Linux file systems

- MINIX
  - up to 64MB system size, 14B file name
- Journal file system (JFS), XFS, etc
- Extended file system (ext)
  - ext, 1992, up to 2GB system, 255B name
  - ext2, 1993, up to 4TB system (1KB block)
  - ext3, since 1998, ext2 + journaling
    - journal (low risk), ordered, writeback (high)
  - ext4, since 2008, 16TB (file), 1EB (system)

These lectures

- File system design and implementation
  - file system structures
  - directory implementation
  - block allocation
  - free-space management
  - consistency and recovery
- Explore further
  - /bin/mount; /sbin/fsck; /etc/exports

Next lectures

- Memory management
  - Main memory
  - Virtual memory