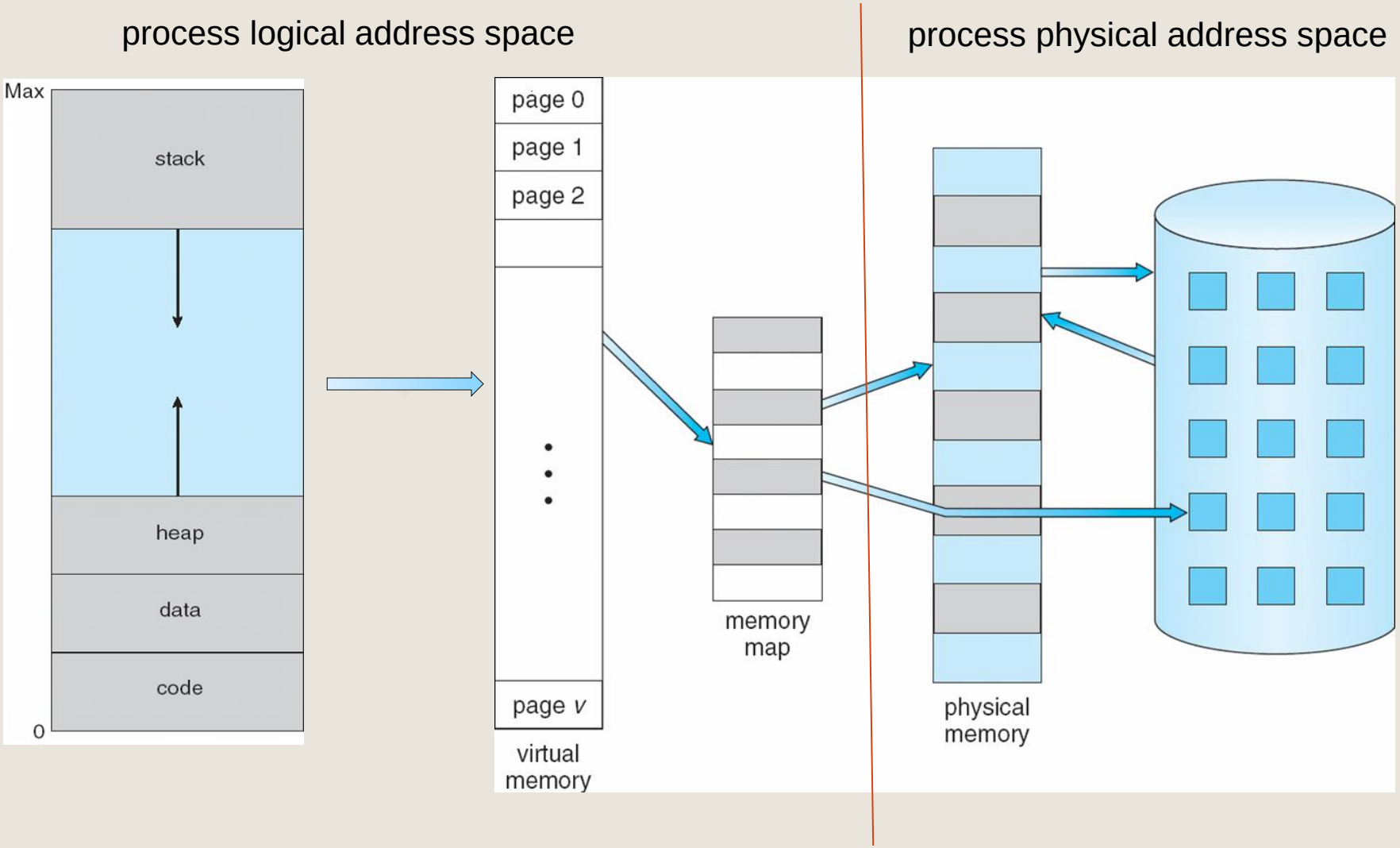
# Operating System Principles - 3

Shyan-Ming Yuan
CS Department, NYCU
smyuan@gmail.com

# Virtual Memory Management

- Virtual memory is a technique that separates user logical address space from physical memory.
- When executing a program, only part of logical address space needs to be in physical memory for execution.
- Logical address space can therefore be much larger than physical address space.
- Allows physical memory to be shared by several processes.
- Allows using less memory for more efficient process creation.
- Demand paging is a most common virtual memory implementation.

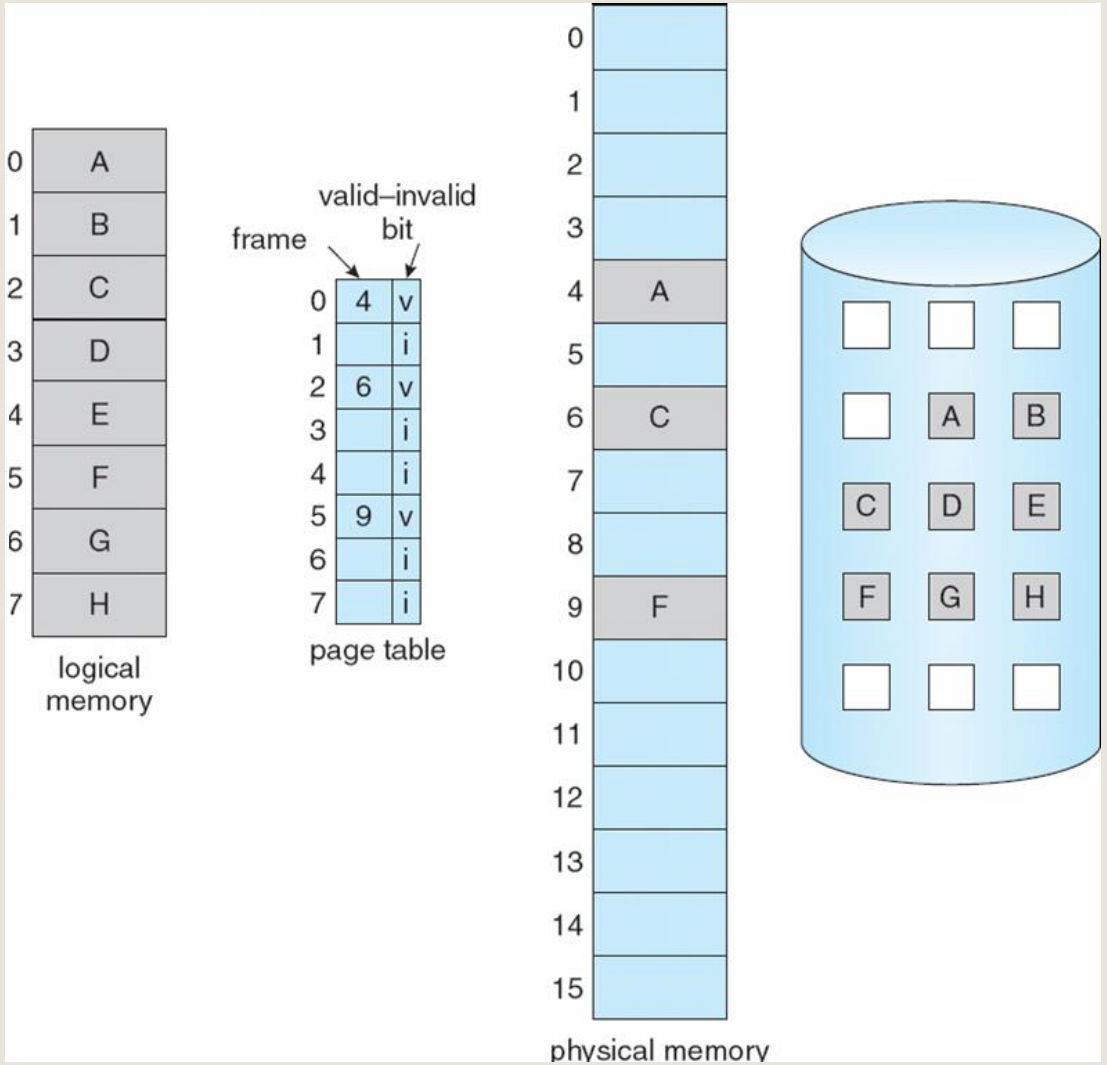
# Virtual Memory > Physical Memory



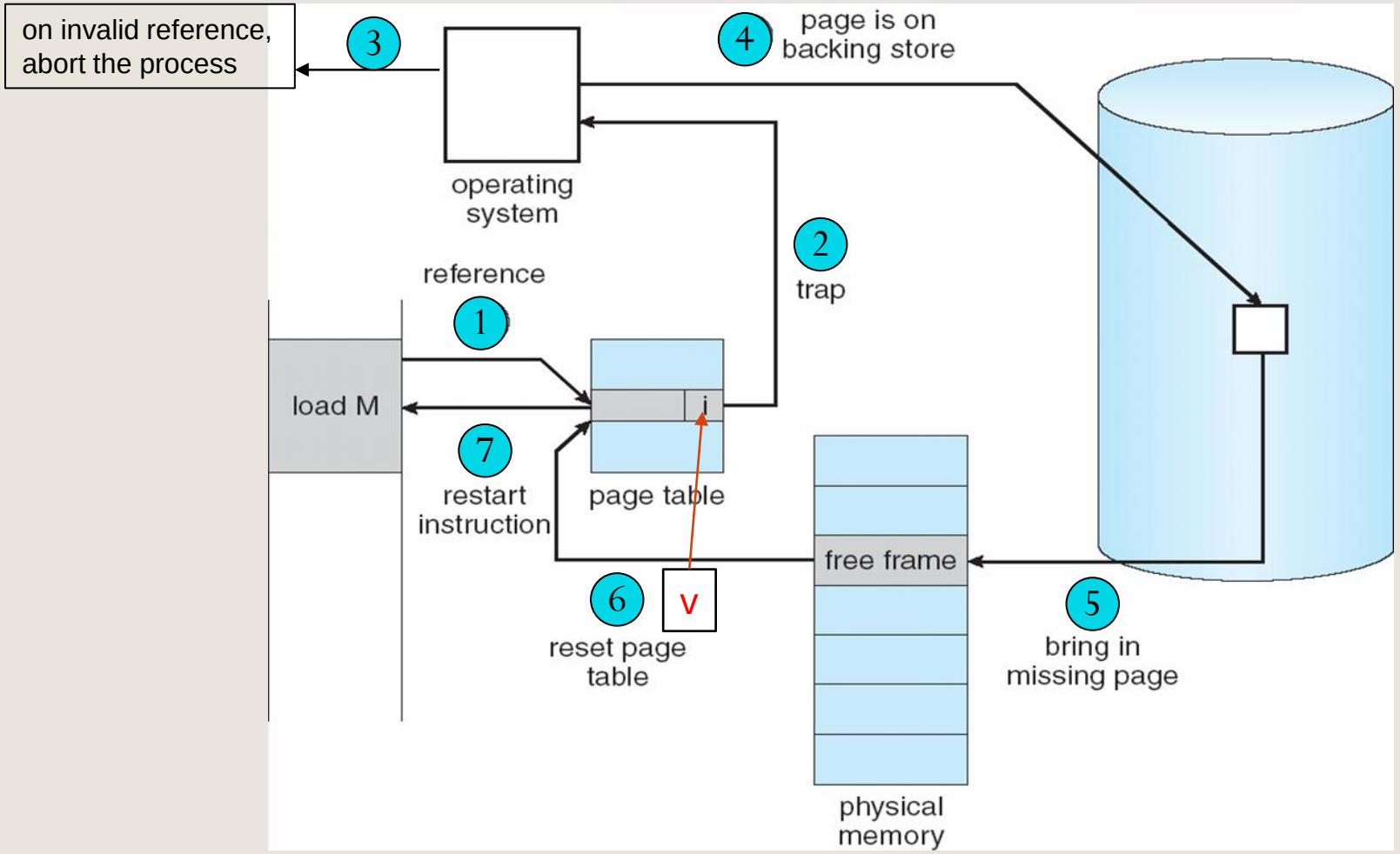
# Demand Paging

- Bring a page into memory only when it is needed.
- Use valid bit in page table entry
  - Valid: the page is in memory ("valid" mapping)
  - Invalid: invalid reference or page is not in memory
    - Trap to OS for page fault exception, the OS check PCB to find out
      - Invalid reference raises a out-of-bound exception and abort the process.
      - Not-in-memory swaps in the missing page.
- Lazy swapper never swaps a not-needed page into memory.
  - A swapper that deals with pages is also called a pager.

# Address Space of A Process



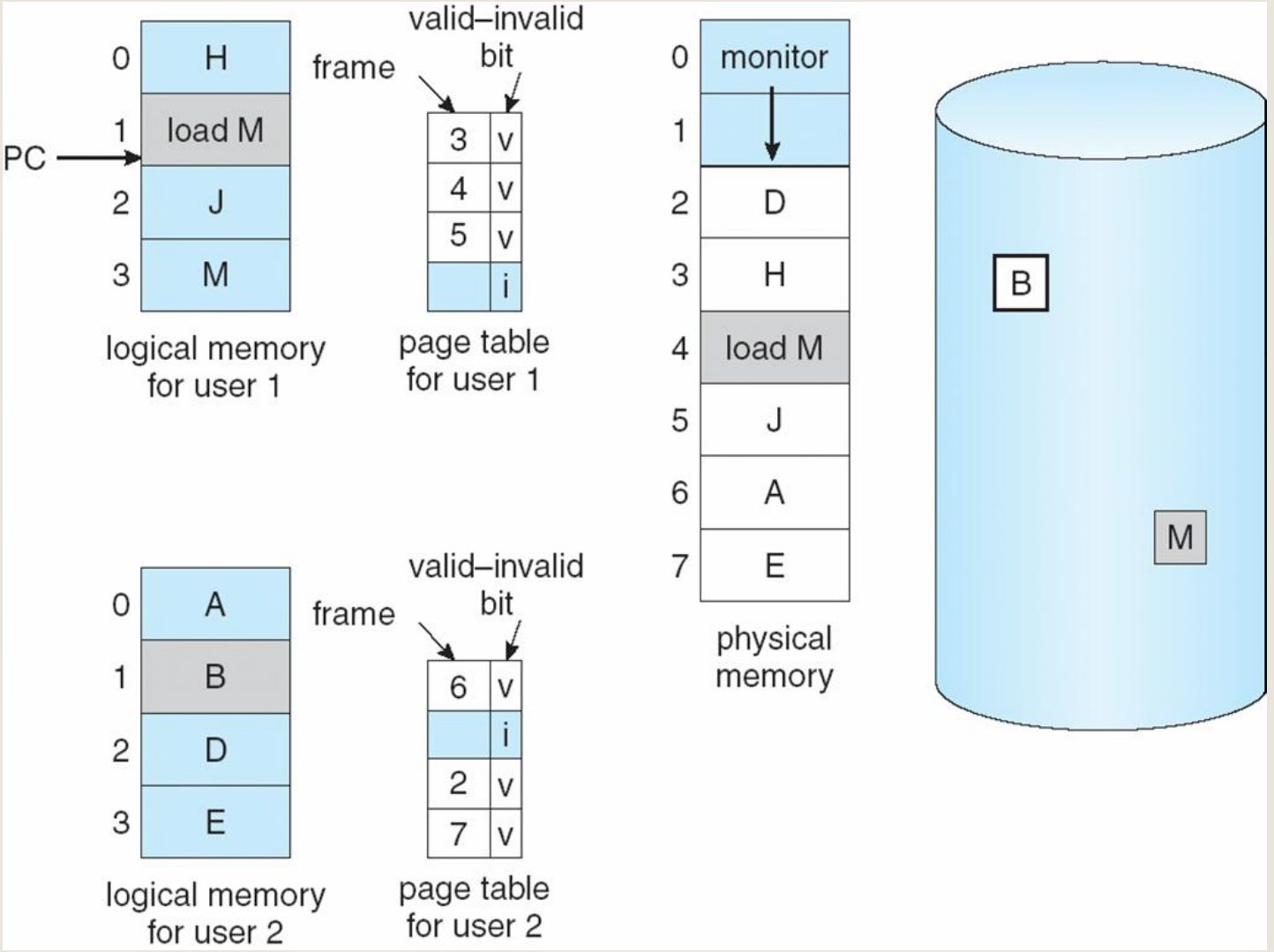
# Page Fault Handling Steps



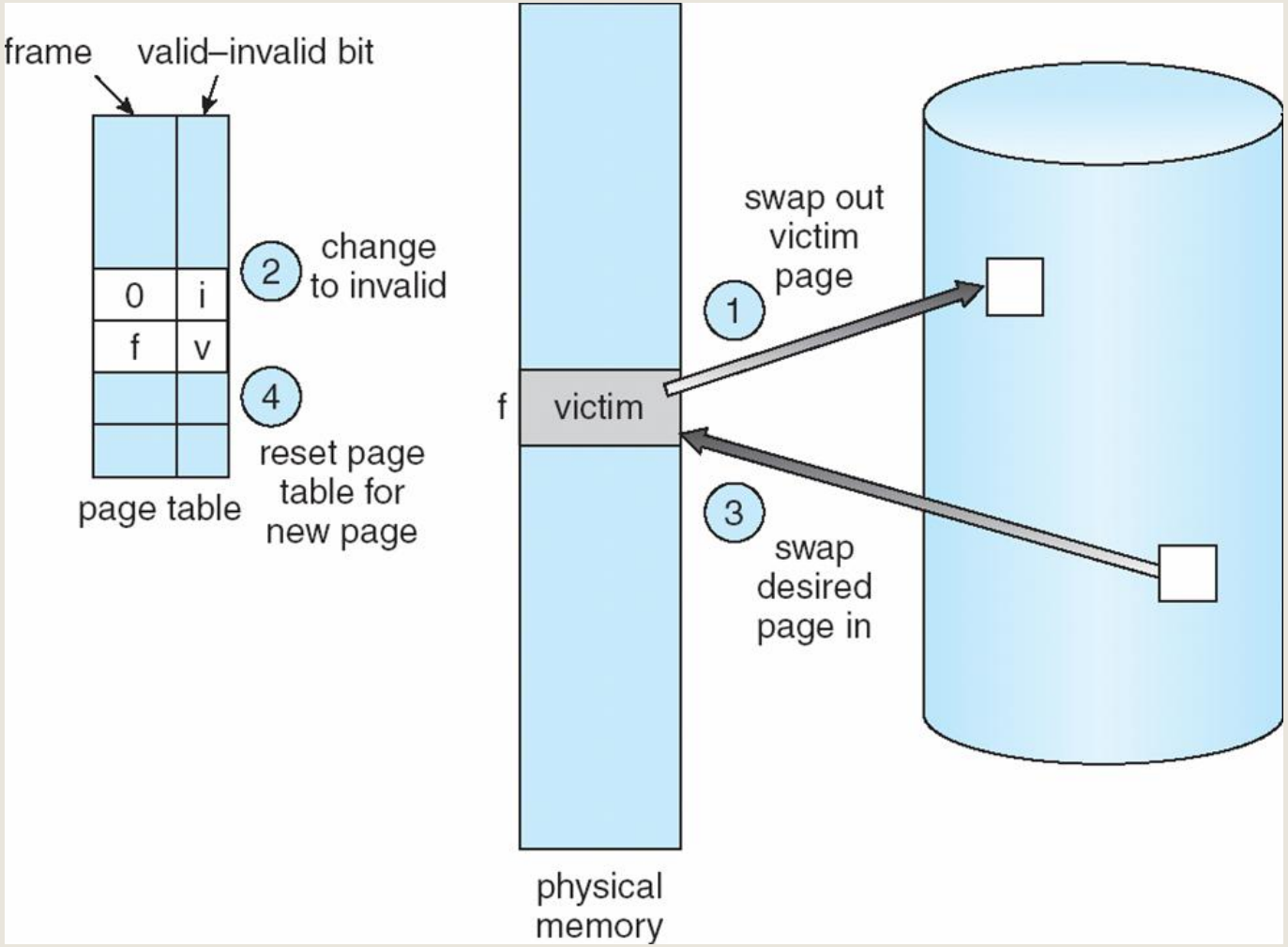
# Page Replacement

- What happens if there is no free frame to page in the needed page?
- Page replacement is the process of finding a page to page out onto disk to create a free frame for the needed page.
  - The main objective of a page replacement algorithm is to minimize the total number of paging activities.
    - Paging is to shuffle page contents between memory and the disk.
  - It is common to use modify (dirty) bit to reduce overhead of paging.
    - Only modified pages need to be written back (page out) to disk.

# Need For Page Replacement



# Page Replacement Steps

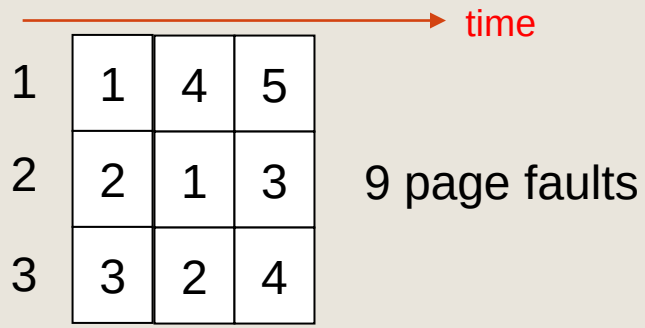


# Page Replacement Algorithms

- First-In-First-Out (FIFO) Algorithm:
  - Replace the page that exists in the memory for the longest time.
- Optimal Algorithm:
  - Replace the page that will not be used for the longest period of time.
  - It is an ideal case if the reference sequence is known in prior.
- Least Recently Used (LRU) Algorithm:
  - Replace the page with the farthest time from the last reference.
- The clock algorithm (also known as the second chance algorithm:)
  - Arrange in-memory pages as a circular list.
  - On selecting a victim, it starts at its current position and looks for pages whose referenced bits are 0. As it does this, it sets the referenced bits of pages it skipped over to 0.

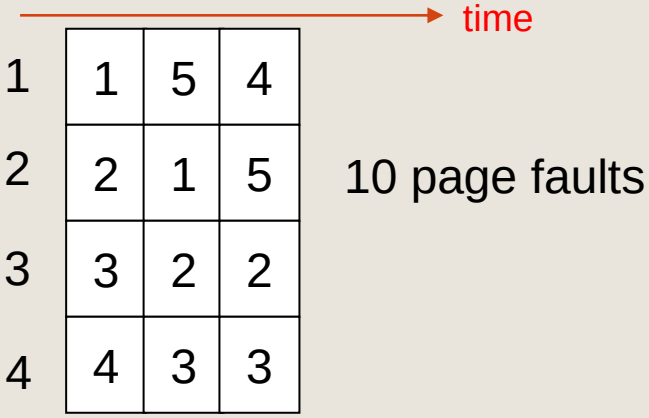
# The Belady's Anomaly in FIFO

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5




9 page faults

- 4 frames: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

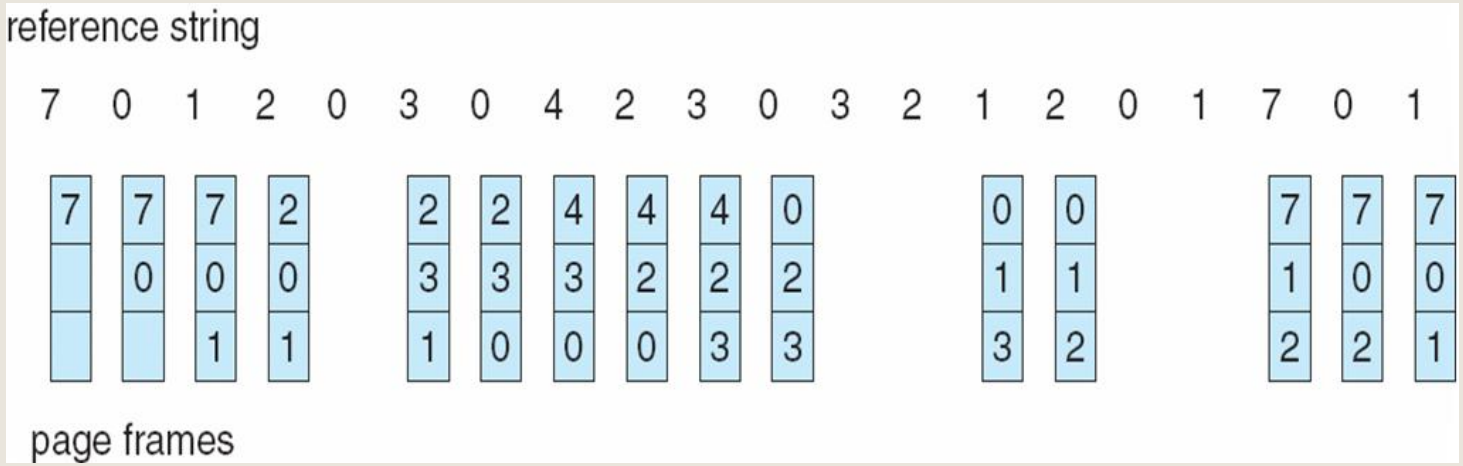



10 page faults

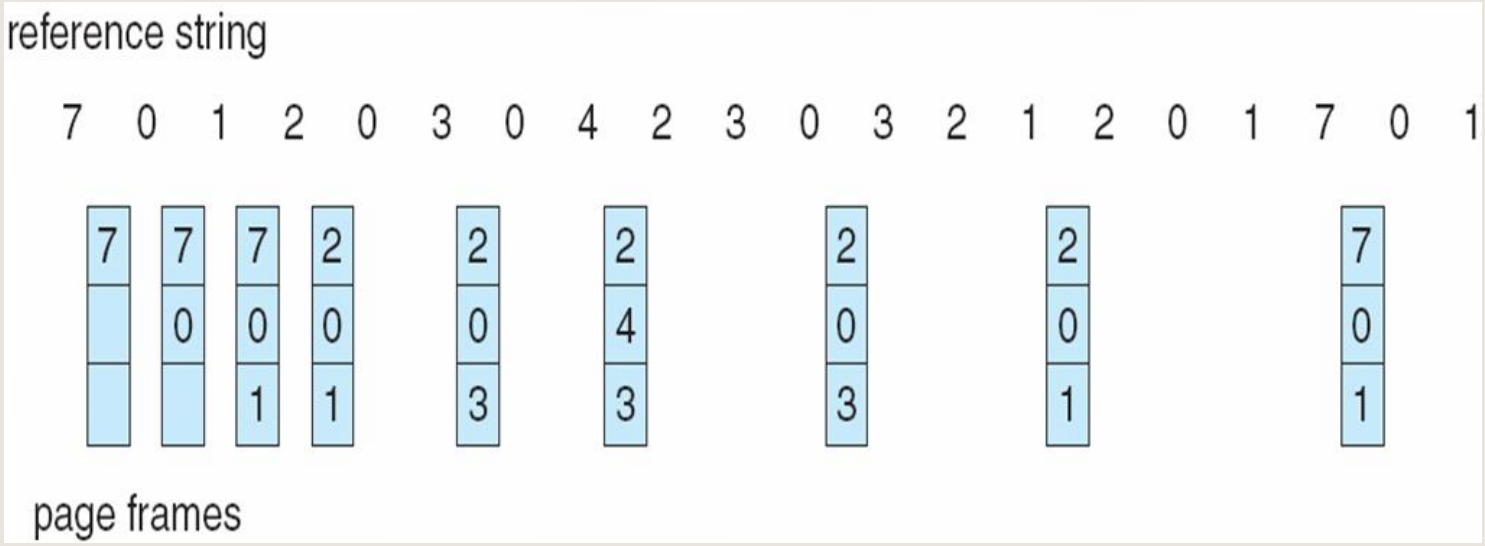
- Belady's Anomaly: more frames => more page faults

# FIFO and Optimal Replacements

The FIFO has
15 page faults

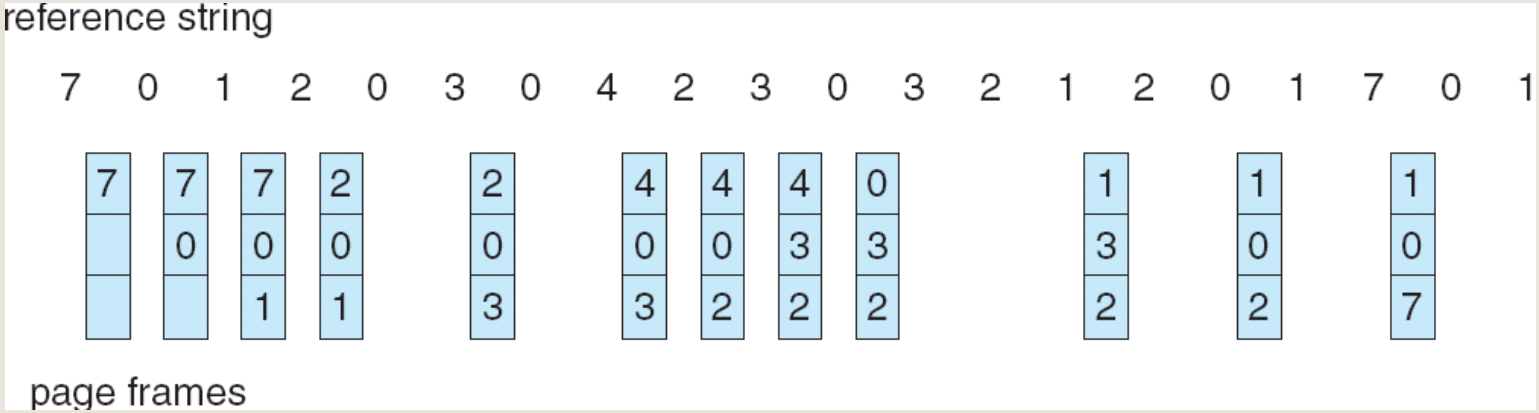



The Optimal has
9 page faults

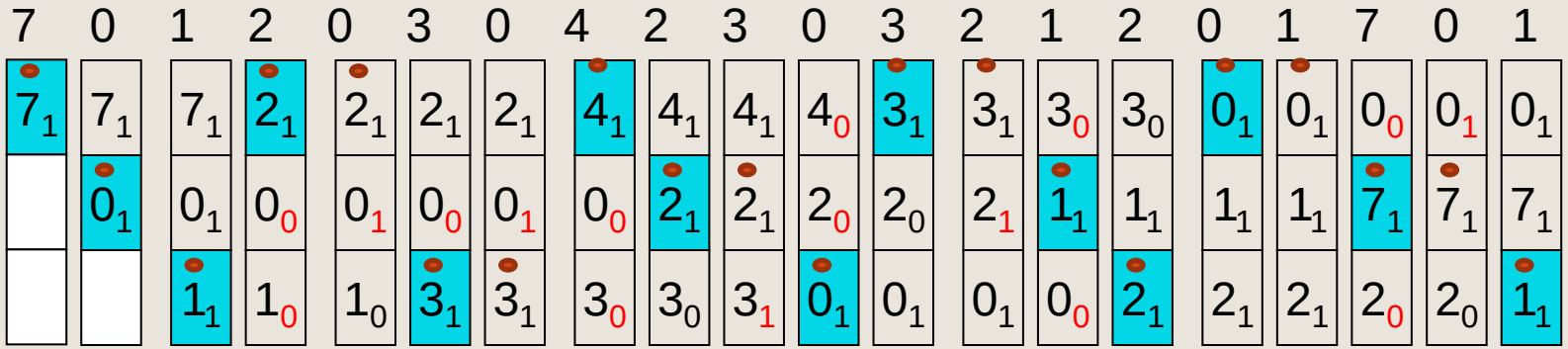


# The LRU & Clock Replacements

The LRU has
12 page faults




The clock has
14 page faults



# How Many Frames for Each Process?

- Each process needs a minimum number of frames for proper instruction execution
  - Example:  IBM 370 needs 6 pages to handle SS MOVE instruction:
    - A 6-byte instruction might span 2 pages,
    - where the from address and to address both might also span 2 pages
- Fixed allocation:
  - Equal allocation assigns each process same number of frames.
  - Proportional allocation assigns frames based on the size of each process.
- Priority allocation is a proportional allocation of priorities.
  - It may select replacement frames from lower priority processes as well.
- Local replacement: each process selects from only its own set of allocated frames.
- Global replacement: a process may select a replacement frame from the set of all memory frames.

# Locality Principle

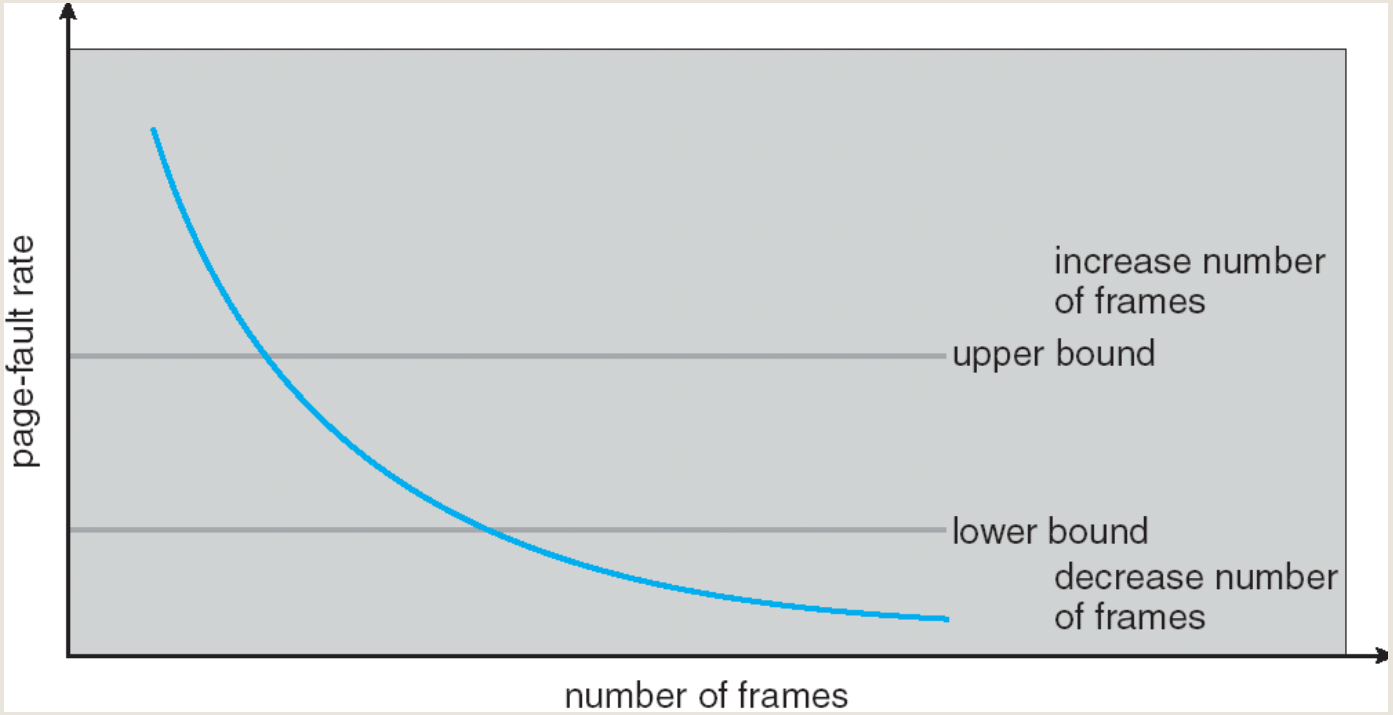
- In general, a process tends to reference the same set of pages over and over before moving on to another set of pages.
  - This set of pages that is currently used is known as the process's working set.
  - Some processes may have bigger working sets than others.
- A process's resident set is the set of pages that is currently in physical memory frames.
- If a process's working set is not a subset of it's resident set then the system will exhibit frequent page swapping between the physical memory and the backing store.
  - It is also called thrashing.
- Thrashing is defined as that a process is busy swapping pages in and out. This leads to:
  - low CPU utilization =>
  - OS will increase the degree of multiprogramming to boots CPU utilization =>
  - more processes are added to the system=> more page faults => severer thrashing

# Resident Set Management

- The locality characteristics of a process is utilized by the working set model to make the resident set of a process covering its working set and avoid thrashing.
  - However, its difficult to recognize the working set of a process in prior.
- Another way to manage the resident set is by monitoring page fault frequency.
  - If a process does not have its working set in memory, it will generate page faults.
    - This is an indication that it needs more memory.
  - If a process is never generating page faults, that could be an indication that it has too much memory allocated to it.
  - Thresholds of page fault frequency can be set to decide how many physical frames should be allocated to a specific process.

# Page-Fault Frequency Scheme

- Establish "acceptable" page-fault rate
  - If the actual rate is too low, the process loses a frame
  - If the actual rate is too high, the process gains a frame



# File Systems

- A file is an abstract data type which consists of a contiguous logical address space for storing persistent information.
- A file system is a container for files. It provides the mechanism to
  - organize, create, store, retrieve, and delete information stored in files.
- The actual data maintained by files are stored on block devices: disk (or flash.)
- A file system used on a block device defines the data structures on the disk blocks.
  - It also defines how directories, files, free space information, and permissions are stored.
- Multiple individual file systems can be combined into an abstract view of a single hierarchical name space (directory tree) via a mount mechanism.
- To support multiple file systems transparently, a layer of abstraction called the Virtual File System (VFS) is used in most modern OSs.

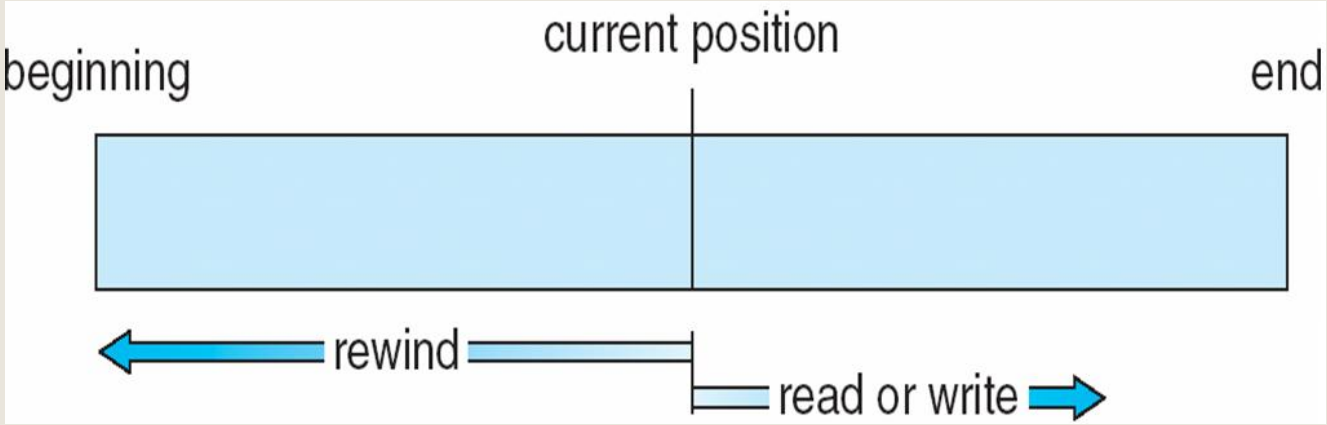
# File Operations

- File is an abstract data type with the following operations:
  - Create and Open
    - Open(Fi) searches the directory structure on disk for entry Fi, and move the content of entry (file attributes of Fi) to memory. The attributes include:
      - File pointer: points to the per-process the last read/write location.
      - Access rights: caches per-process access mode information.
      - File-open count: records the number of times that Fi was opened concurrently.
      - Disk location information: caches disk addresses and related data access information.
  - Write and Read
  - Reposition within file
  - Delete and Truncate
  - Close (Fi) stores the content of entry Fi in memory back to directory structure on disk if necessary.

# File Access Methods

- Sequential Access

  read next
  write next
  rewind




- Direct Access (Random Access) *n* is a relative block number

  read n
  write n
  position to n

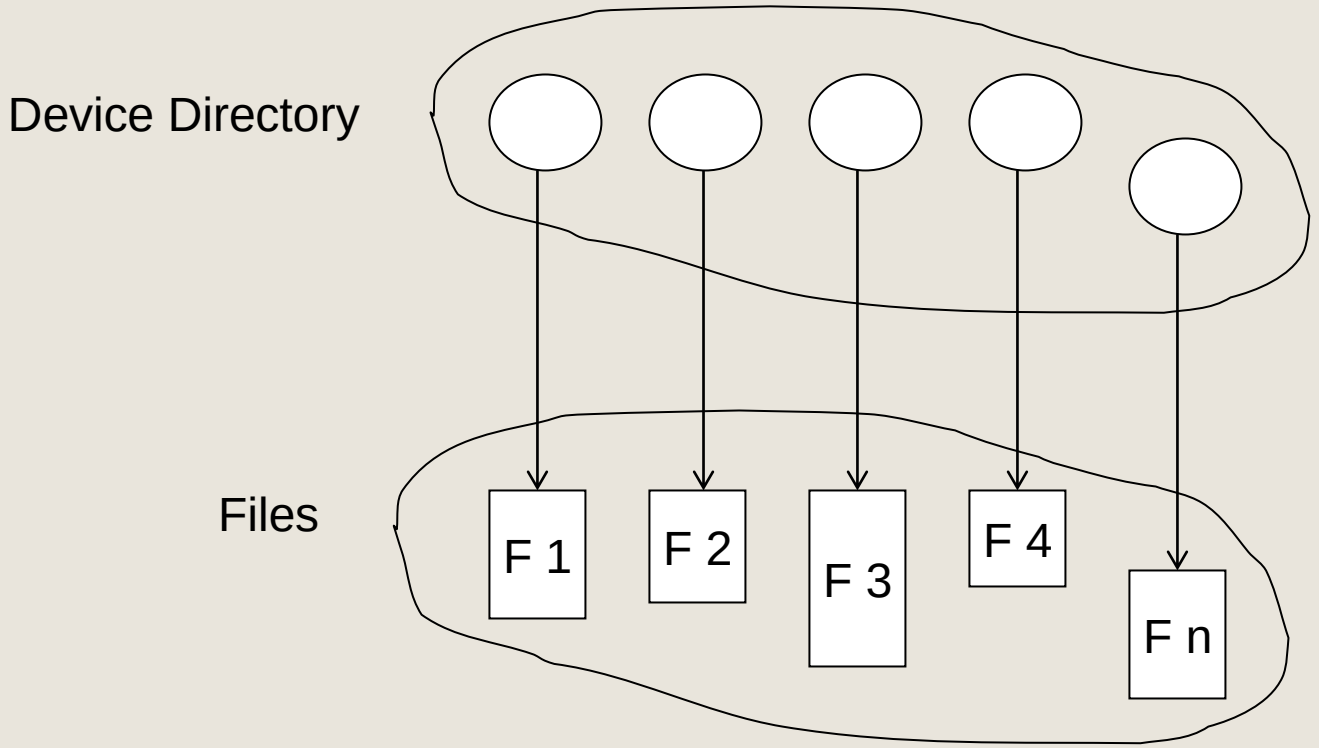
| Sequential access | Simulated by direct access |
|---|---|
| rewind | cp := 0; // position to cp |
| read next | read cp;<br>cp := cp + 1; // position to cp |
| write next | write cp;<br>cp := cp+1; // position to cp |

# Disk Structure

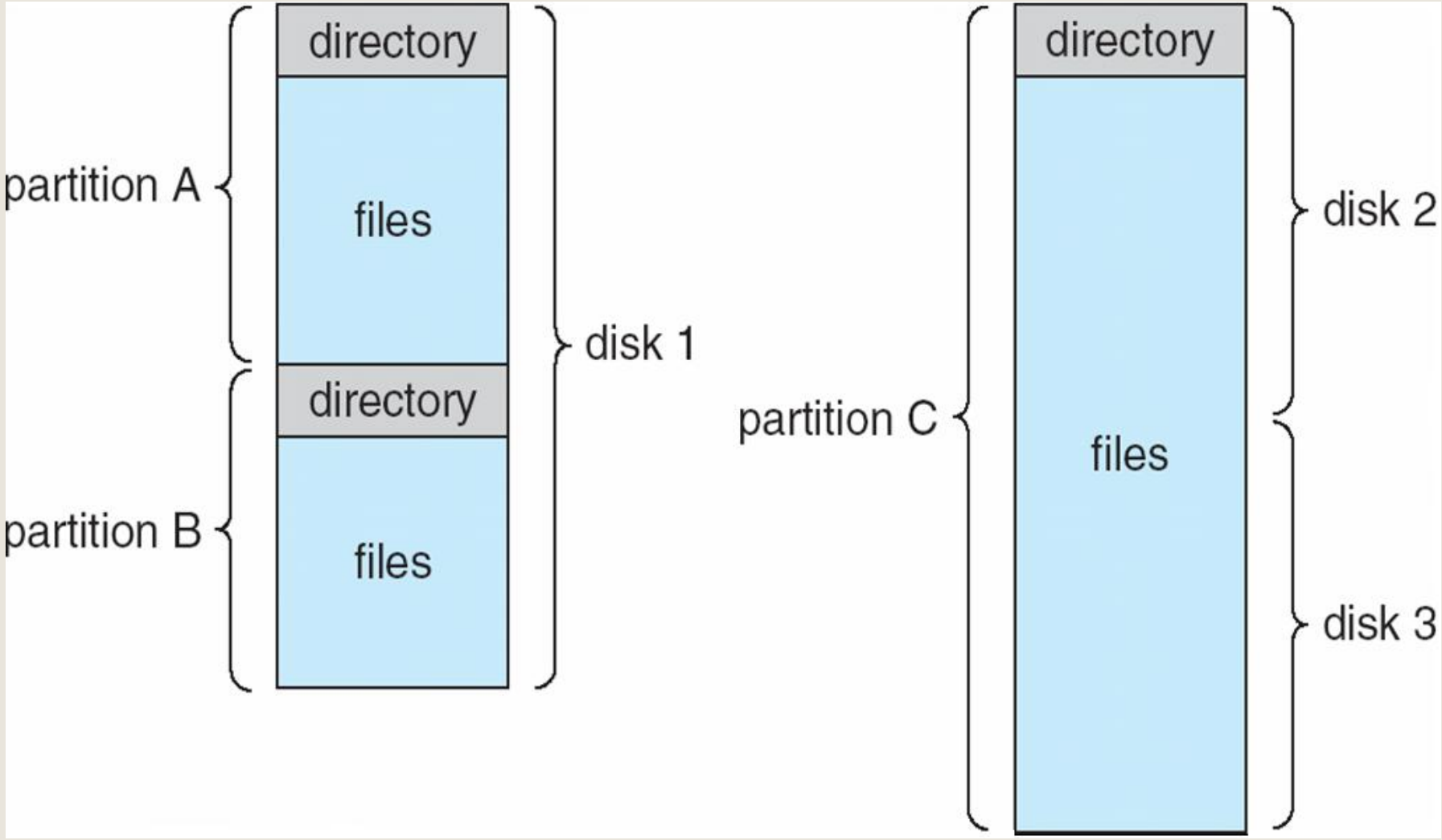
- Disk can be subdivided into partitions.
- Disks or partitions can be RAID protected against failure
- Disk or partition can be used in two modes:
  - raw without a file system
  - formatted with a file system
- Partitions are also known as minidisks or slices.
- An entity containing a file system is known as a volume.
  - Each volume contains a file system.
  - It records that file system's info in device directory or volume table of contents.

# Device Directory Structure

- A device directory is a collection of entries containing information (file attributes) about all files stored in the device.



# A Typical File-system Organization

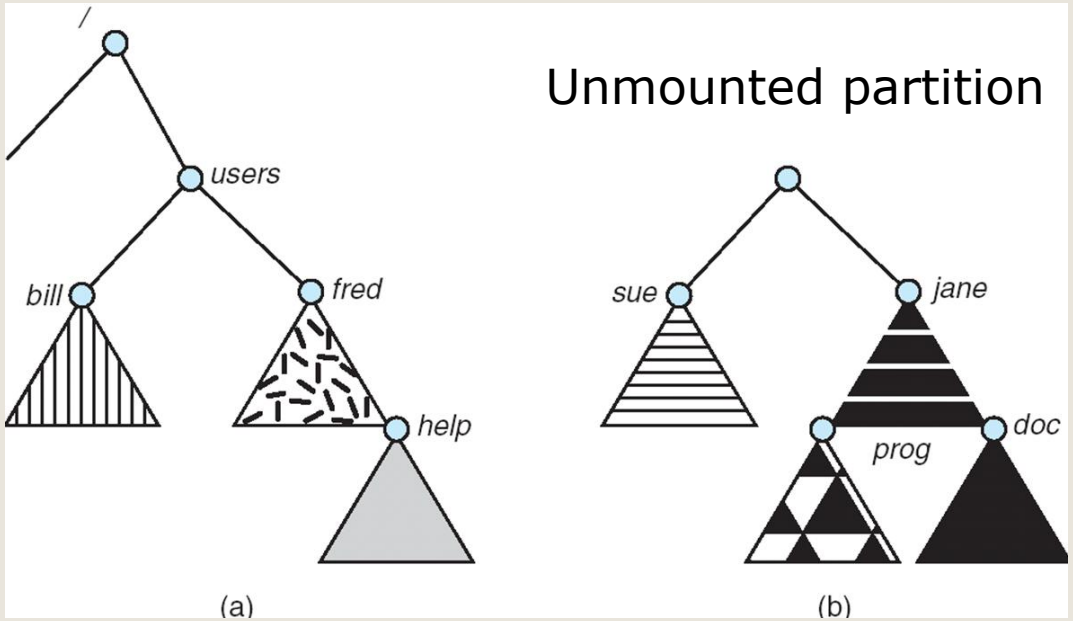


# File System Mounting

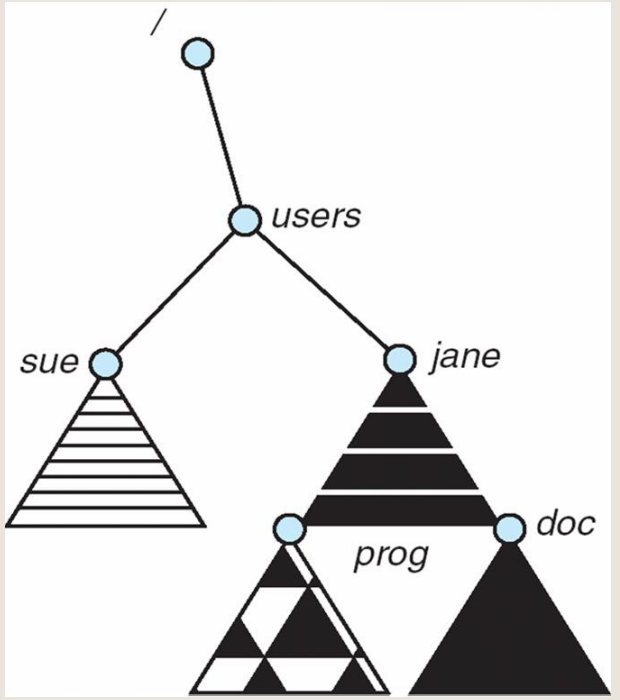
- A file system must be mounted before it can be accessed
- A unmounted file system is mounted at a mount point

Existing file system

Unmounted partition




After mounted in /users

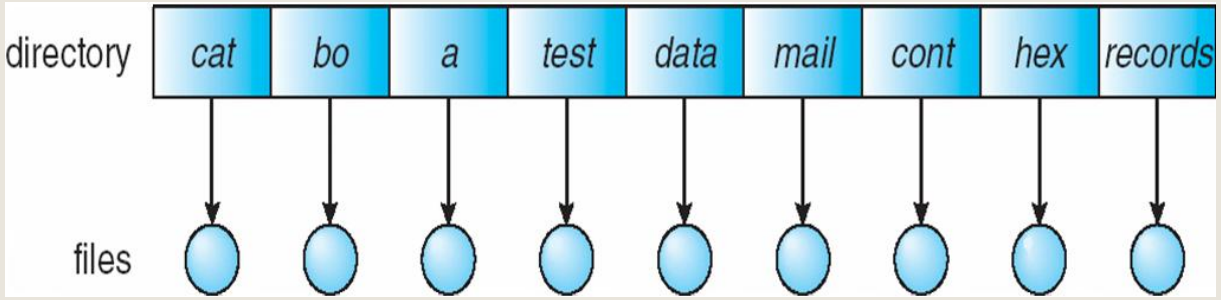


# Directory is a Special File

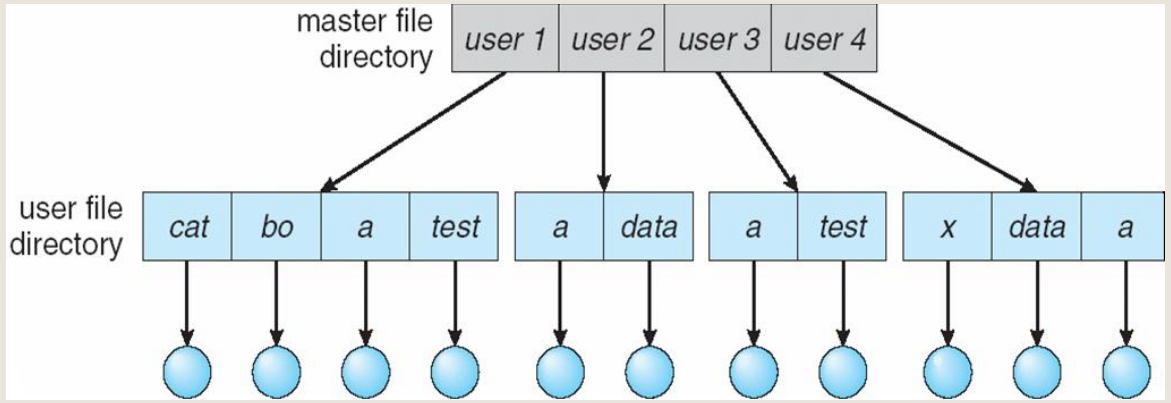
- Directory operations:
  - Search a file from a directory
  - Create and Delete a file entry from a directory
  - List all files from a directory
  - Rename a file
  - Traverse the directory hierarchy
- Advantages of directory:
  - Efficiency – locating a file quickly
  - Naming – convenient to users
    - Two users can have same name for different files
  - The same file can have several different names
  - Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

# Types of Directory

- Single-Level Directory: a single directory for all users (MS-DOS)
- Two-Level Directory: separate directory for each user (early time sharing systems)
- Tree-Structured Directories: subdirectories can be created by users dynamically
- Acyclic-Graph Directories: subdirectories and files can be shared
- General Graph Directory: most general and difficult to manage

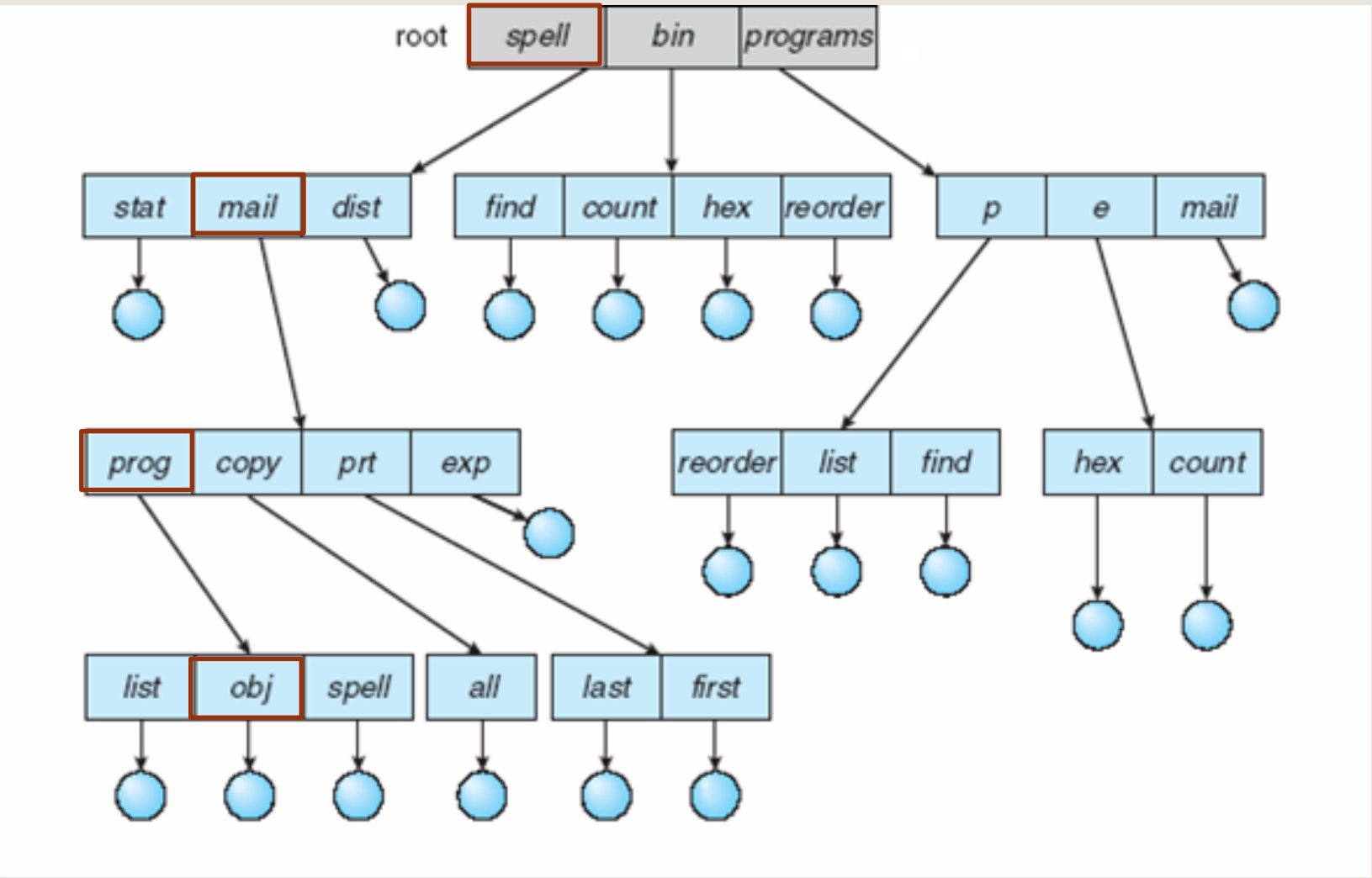



Single-Level Directory




Two-Level Directory

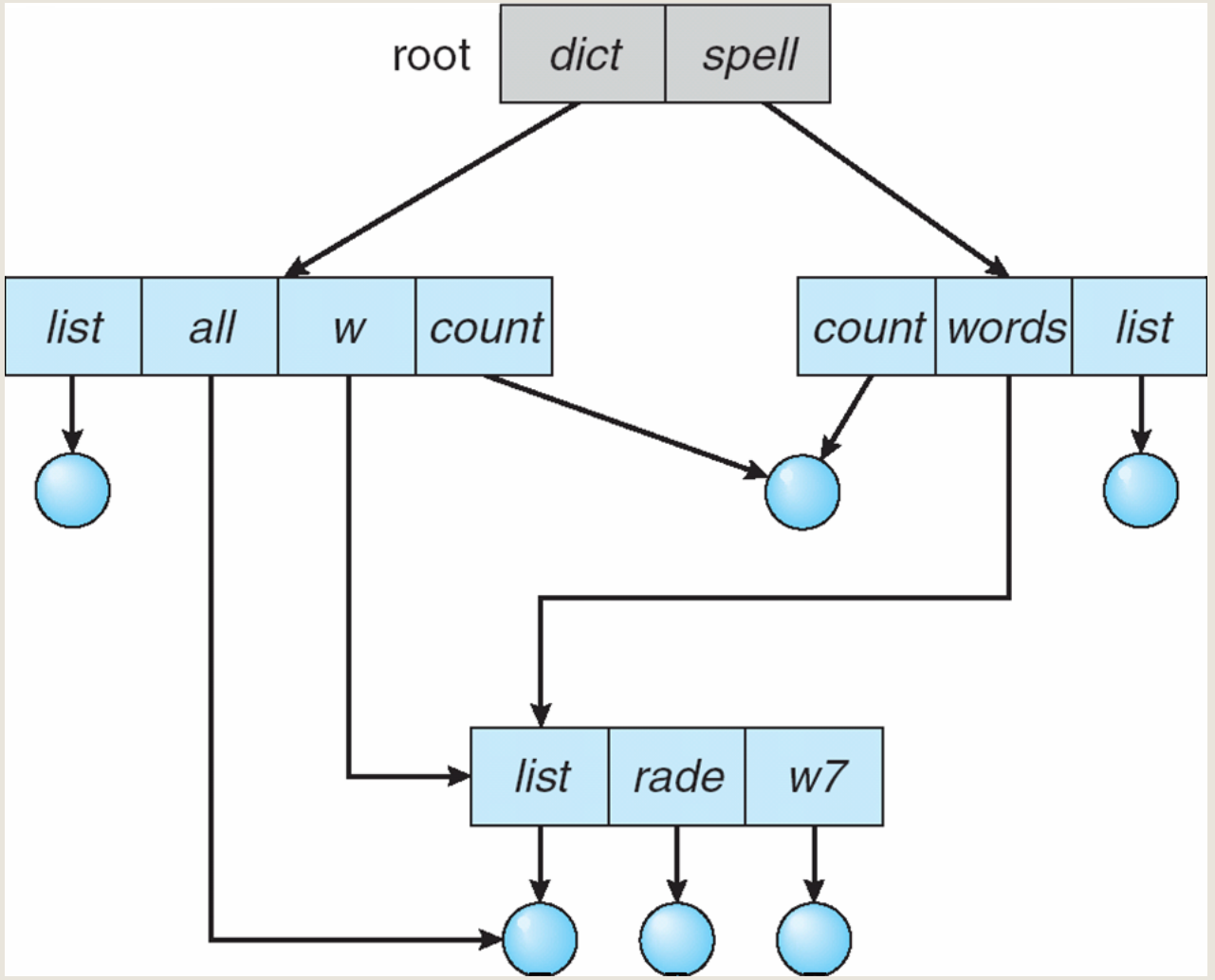
# Tree-Structured Directories




- Efficient searching
- Grouping Capability
- Absolute or relative path name
  - cat /spell/mail/prog/obj (absolute)
- Current directory (working directory)
  - cd /spell/mail/prog
  - cat obj (relative path name)

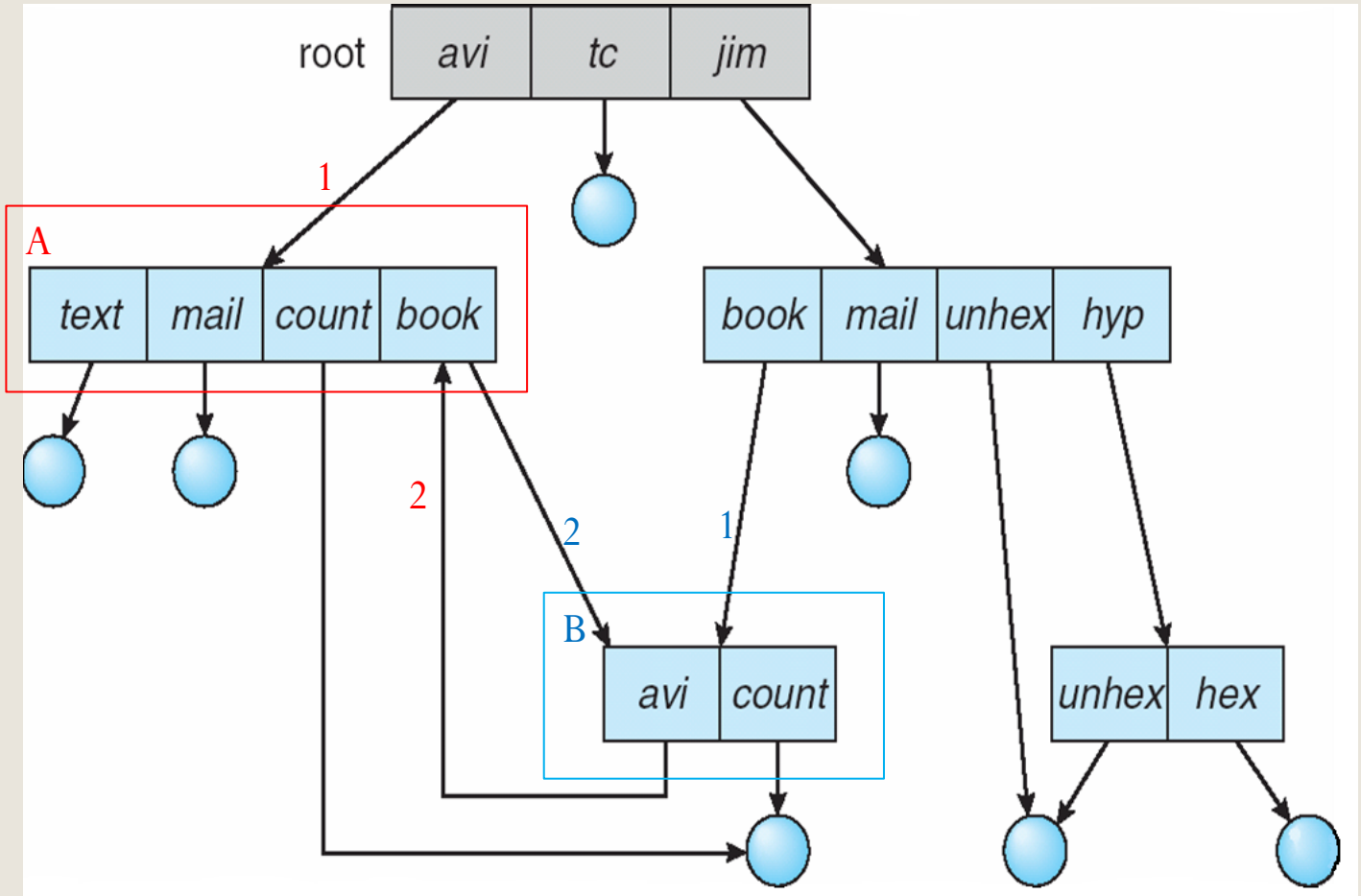
# Acyclic-Graph Directories

- A file can have two different names (aliasing)
- If deletes the file /dict/w/list then
  - The file /dict/all may be lost
- Possible solutions:
  - Reference count records # of names: Hard Link
  - Uses a new directory entry type
    - Soft Link: another name (pointer) to an existing file
      - A special file containing the real path name of the original file
      - Resolve the link: follow pointer to locate the file
- On deletes /dict/w/list,
  - Only the reference count is decreased by 1.
  - If the reference count becomes 0,
    - then it can be removed from the file system.



# General Graph Directory

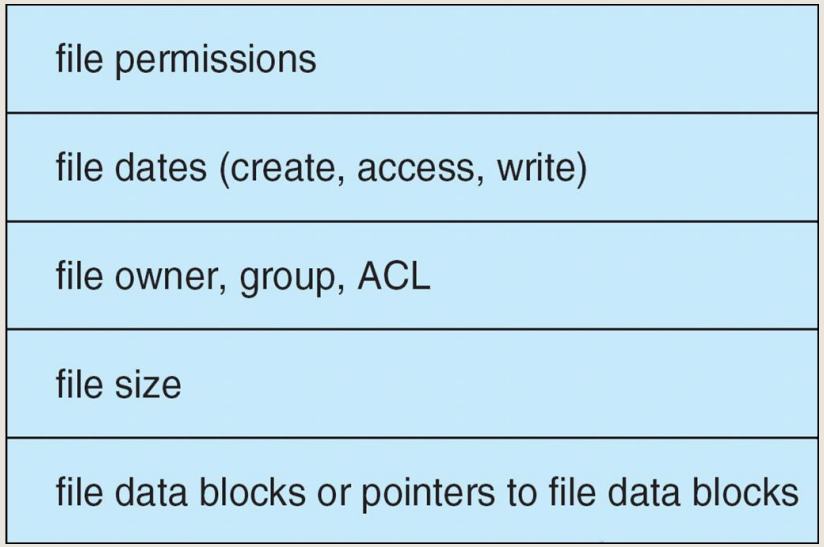
- Cycle will cause reference count invalid.
  - On creates /avi/book as /jim/book
    - Count(B) ← 2
  - On creates /jim/book/avi as /avi
    - Count(A) ← 2
  - After deletes /jim/book and deletes /avi/book
    - Count(B) is 0 and deleted, where count(A) is 2 but A has only one valid name "/avi" .
- To avoid generating cycles
  - hard links to files not subdirectories
  - Garbage collection
  - Before adding a new link,
    - Check cycle existence.
  - Soft link instead of hard link.



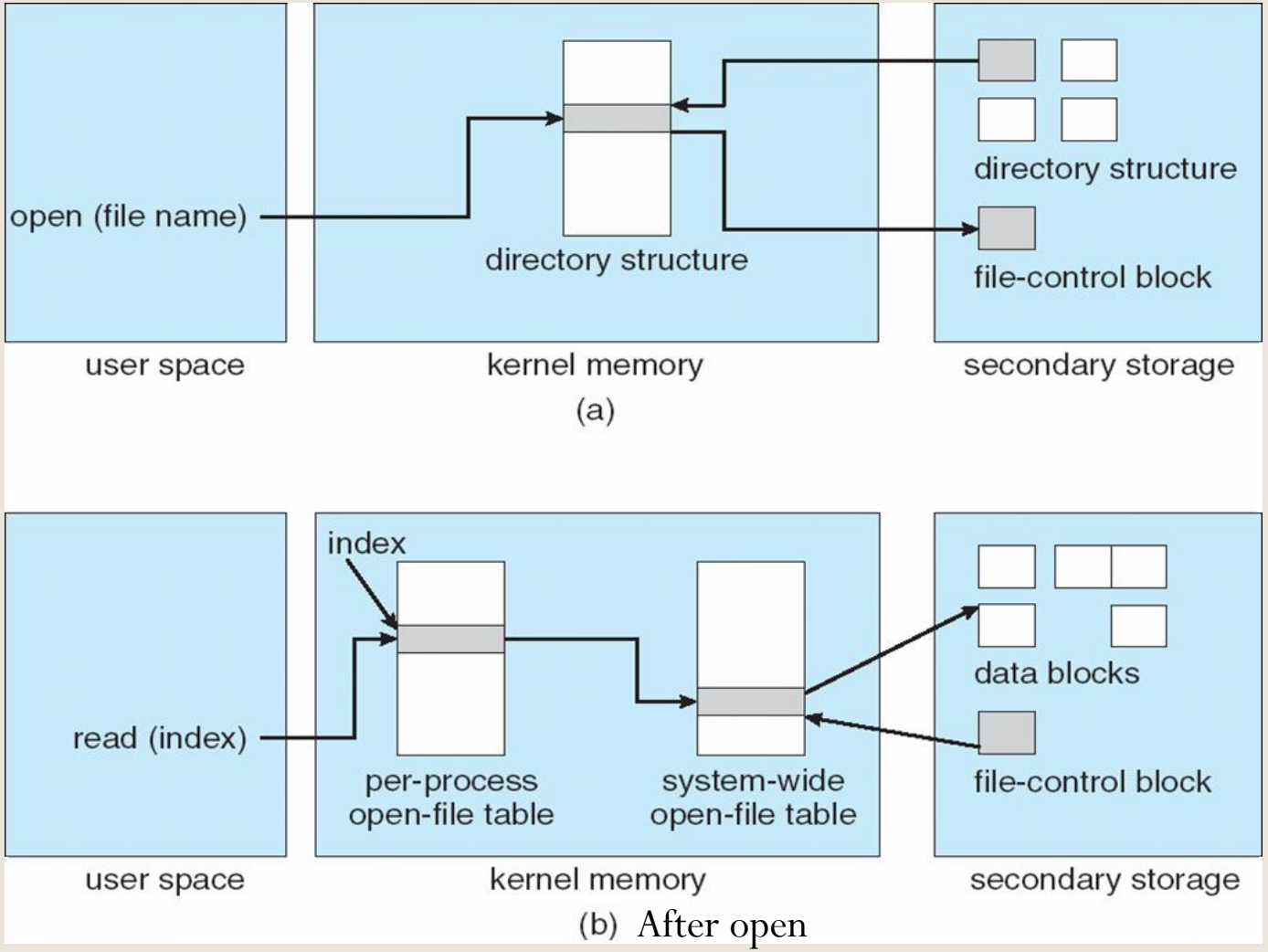
# File-System Implementation

- A file system usually resides on secondary storage (disks)
  - It provides easy access to disk by allowing data to be stored, located, and retrieved easily.
  - It uses per-file file control block (FCB) to store meta information about a file.
- Device directory is used to maintain directories and files and store their FCBs.

A typical file control block (FCB)

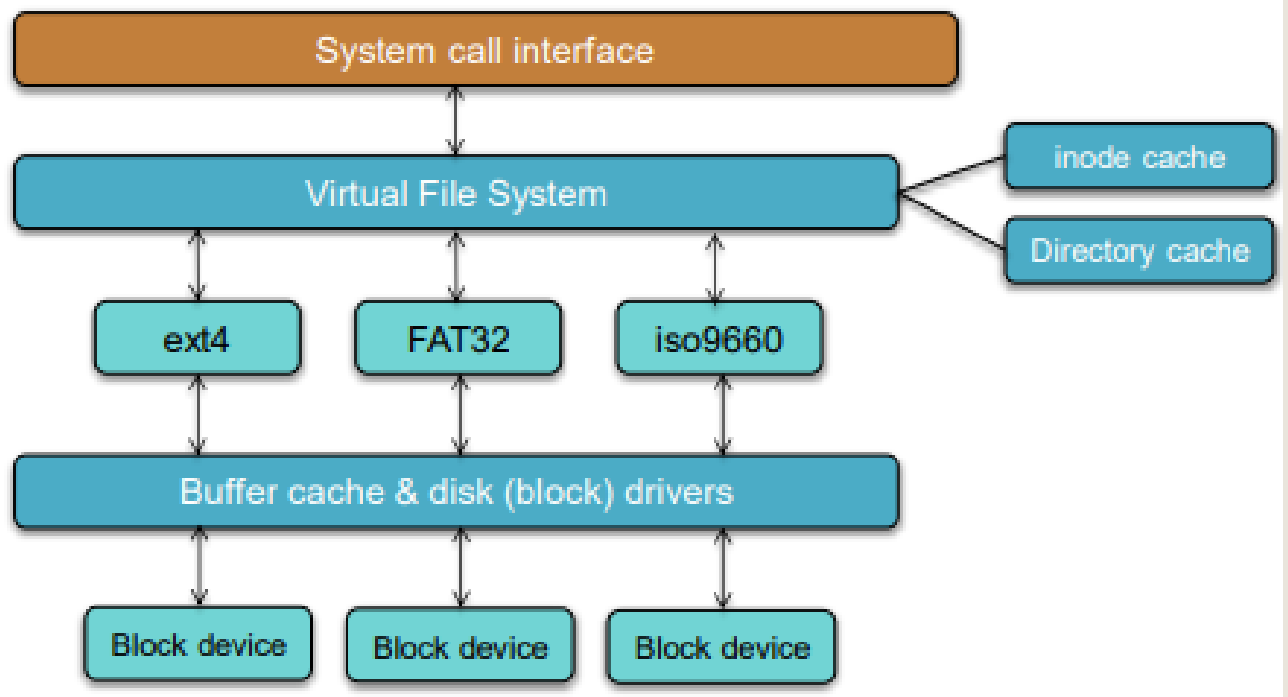


# In-Memory File System Structures used in OS



# Virtual File System (VFS) Interface

- Virtual File Systems (VFS) provide an abstract interface for file system implementations.
  - VFS allows the same system call interface (the API) to be used for different file systems.
- The API is to the VFS interface, rather than any specific type of file system.
  - Each real file system interface exports the common VFS API.



# Directory Organization

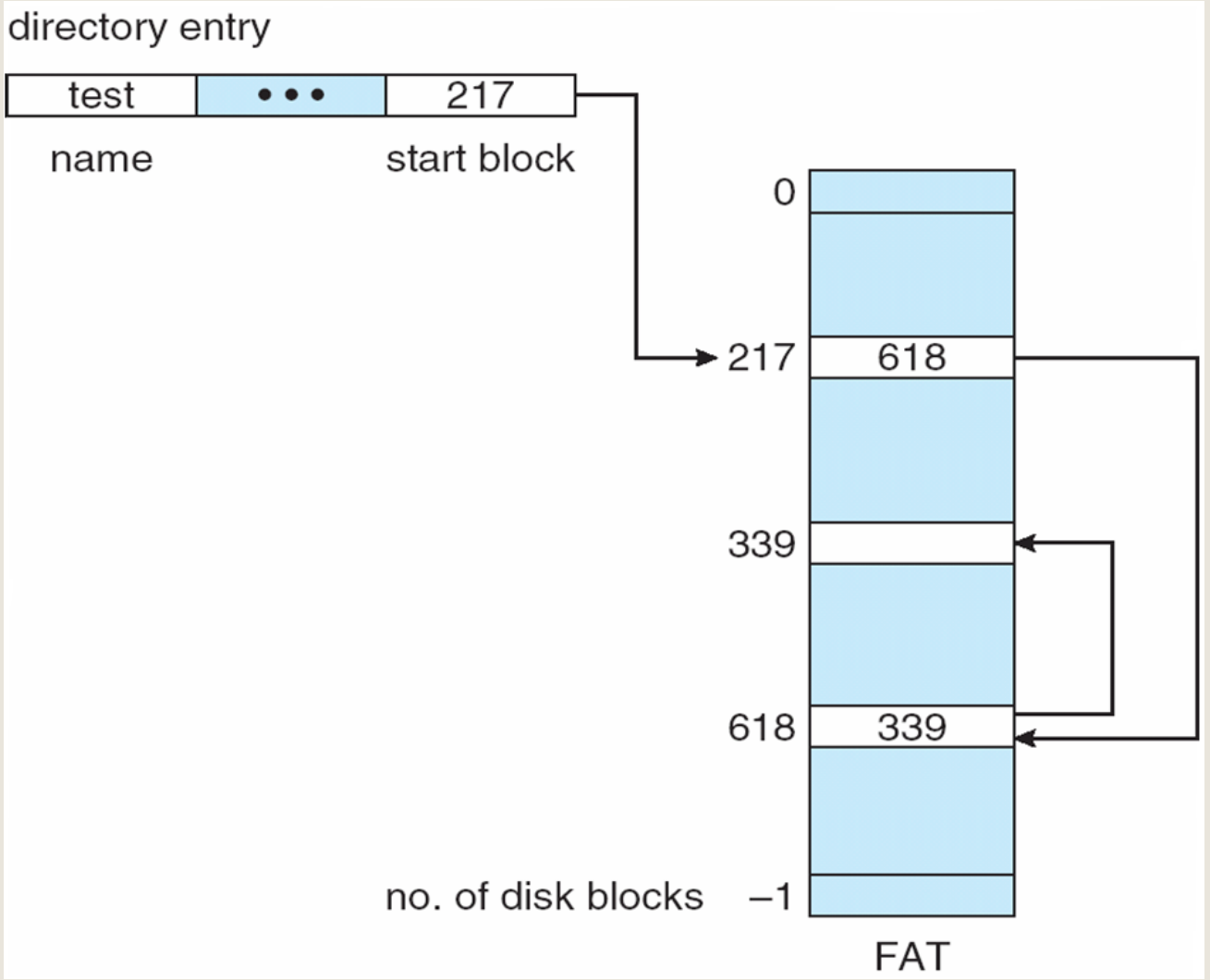
- A directory is just a file containing entries of (name, reference) pairs.
  - Name → (metadata, data) Unix (UFS) approach
  - (Name, metadata) → data MS-DOS (FAT) approach
- These entries can be organized either as a Linear list or a Hash Table.
  - In linear list, searching a large directory is very slow.
    - It may be improved by caching frequently-used entries.
  - In linear list with a hash table, directory search time can be decreased.
    - Hash collisions and fixed hash table size need to be considered.
- More complex structures may be used.
  - B-Tree, Htree, and etc..
    - Balanced tree with constant depth

# Block Allocation Methods

- Disk blocks may be allocated to a file by the following methods.
  - Contiguous allocation: assigns each file a set of contiguous blocks on the disk.
    - It is a simple mechanism. Only the starting location (block #) and length (number of blocks) are required.
    - It provides random access to any file position.
    - Problems: files cannot grow easily and storage may become wasteful.
    - Compromise solution: uses extents
      - Allocate a contiguous chunk of space (called extent) at a time.
      - Keep track of all extents information (location, size)
  - Linked allocation: links a file's data blocks into a linked list of blocks.
    - Directory entry contains a pointer to the first block of the file.
    - Each block contains a pointer to the next block.
    - Problems: bad for random access and each block uses space for the pointer to the next block
    - Compromise solution: uses File-allocation table (FAT)
  - Indexed allocation: store all block pointers of a file in one place: the index block (inode)

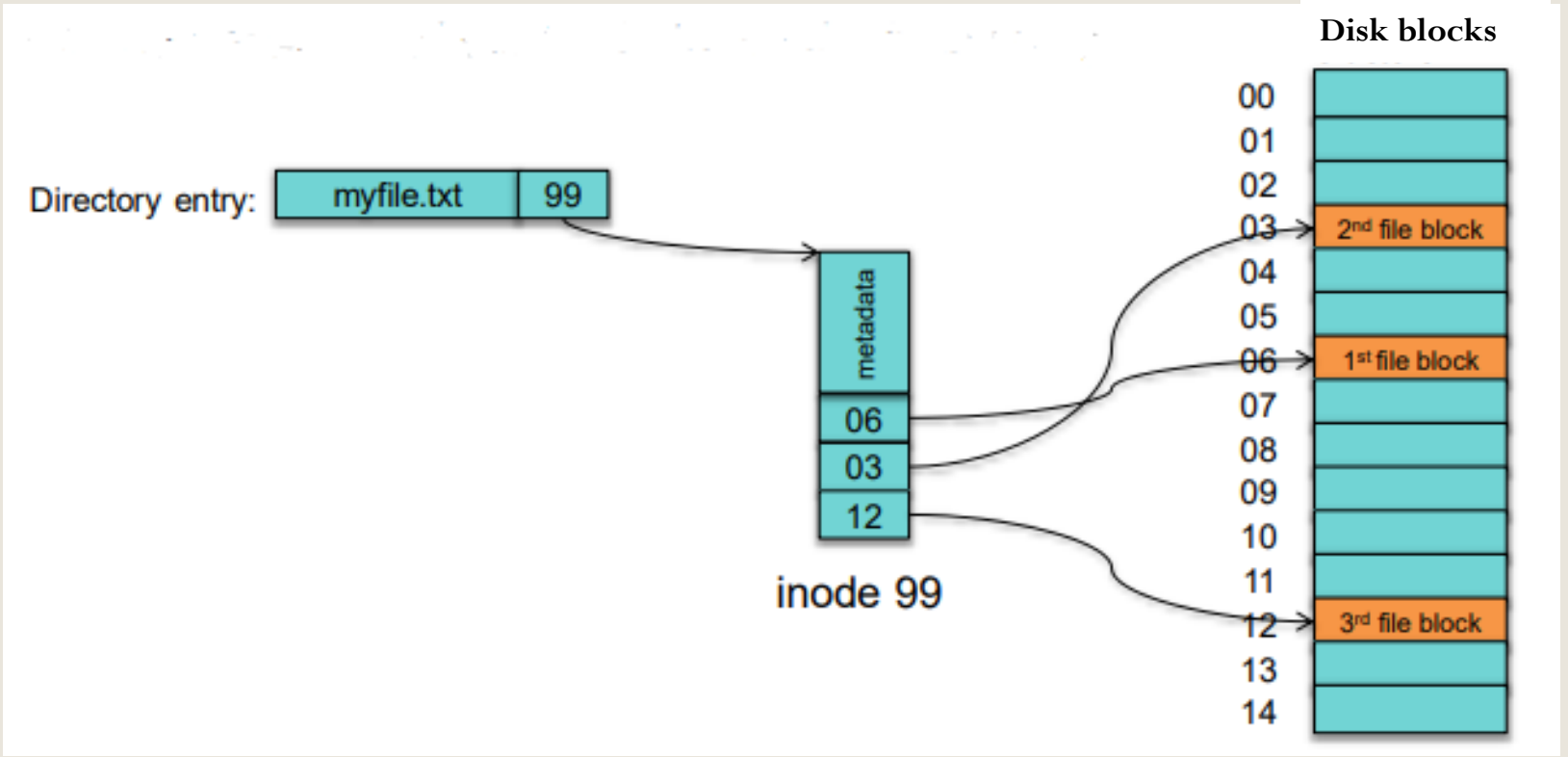
# File-Allocation Table (FAT)

- It is a variation of Linked Allocation
  - used in MS-DOS and early MS Windows
- Each disk volume contains a FAT table that has one entry for each disk block.
  - The FAT is shared by all files stored in the disk volume.
- Each FAT entry contains the disk location of next logical block in the file.
- The directory entry of a file contains the disk location of the 1st logical block.



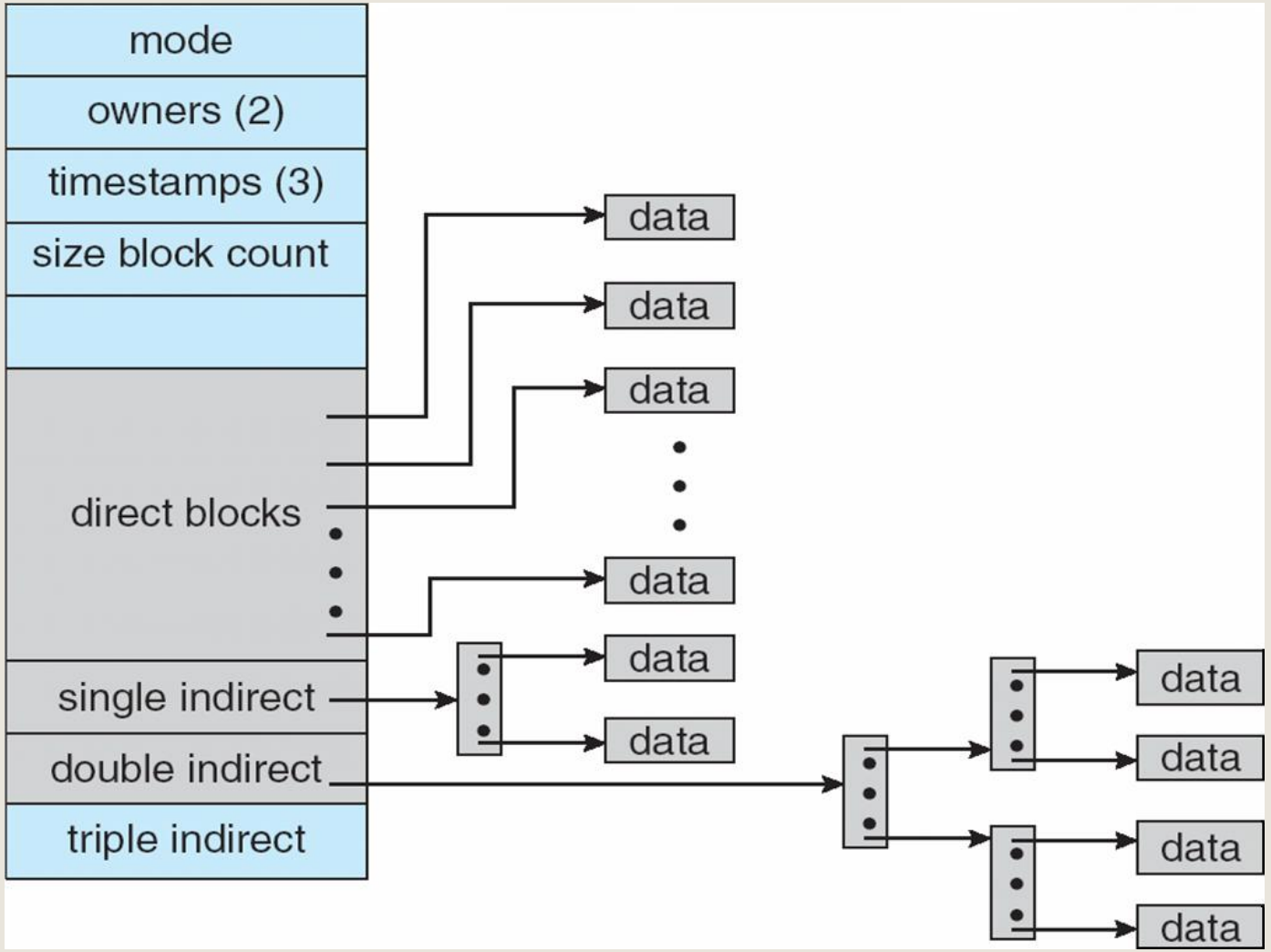
# Indexed Allocation

- Directory entry contains file name and inode number.
  - Each inode contains file metadata (length, timestamps, owner, etc.) and a block map.
  - On file open, read the inode to get the index map



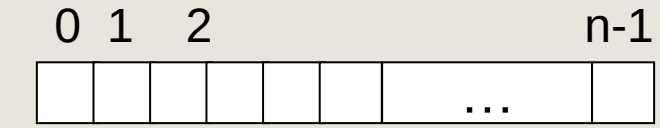
# Combined Scheme: UNIX UFS (4K bytes per block)

- To avoid variable sized inodes.
- An inode may contain different types of block locations.
  - Locations of data blocks
  - Location of single indirect block
  - Location of double indirect block
  - Location of triple indirect block

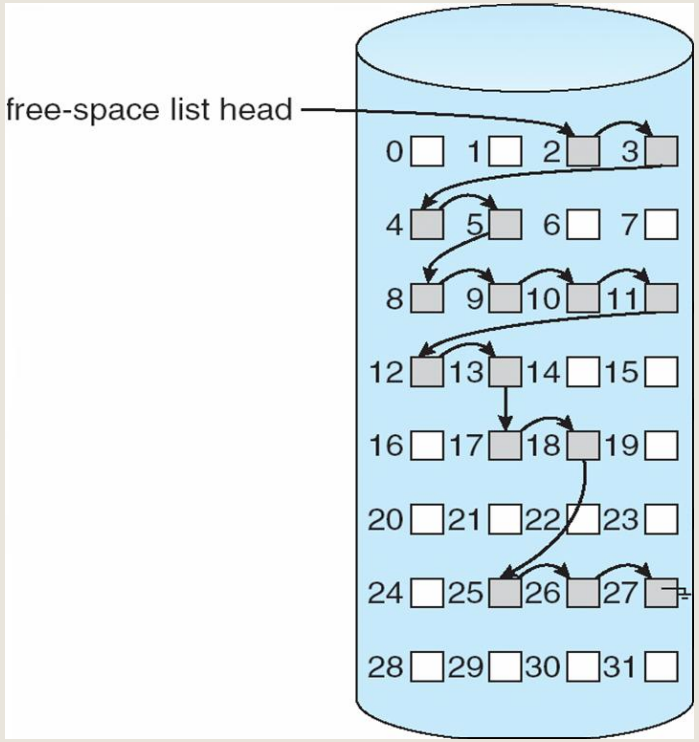


# Free-Space Management for disk blocks

- Bit vector (n blocks) requires extra space
  - Example: 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 :




$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$



# The Grouping and Counting Methods

- The Grouping method is the modification of the linked list method.
  - The first free block (indexing free block) stores the addresses of the next n free blocks.
    - The first n-1 blocks are normal free blocks.
    - The last block is another indexing free block which again stores n addresses of free blocks, and so on.
- The advantages of the grouping method are
  - The addresses of a large number of free blocks can be found quickly.
  - It can allocate a collection of empty disk blocks efficiently.
- The Counting method is also a modification of the linked list method.
  - Each free block in the disk contains two fields:
    - The address of the next free block.
    - The number of free contiguous blocks following it.

# Log Structured (Journaling) File Systems

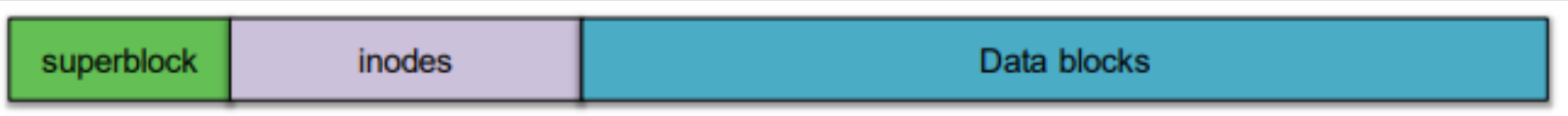
- Journaling file systems record each update to the file system as a transaction.

```
Transaction-begin
  New inode 779
  New block bitmap, group 4
  New data block 24120
Transaction-end
```

- All transactions are written to a log first.
  - 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 then asynchronously written to the file system.
  - When the file system is modified successful, the transaction is removed from the log.
- If the file system crashes, all transactions in the log must be replayed on reboot.
  - It is also called redo logging.

# Unix File System (UFS)

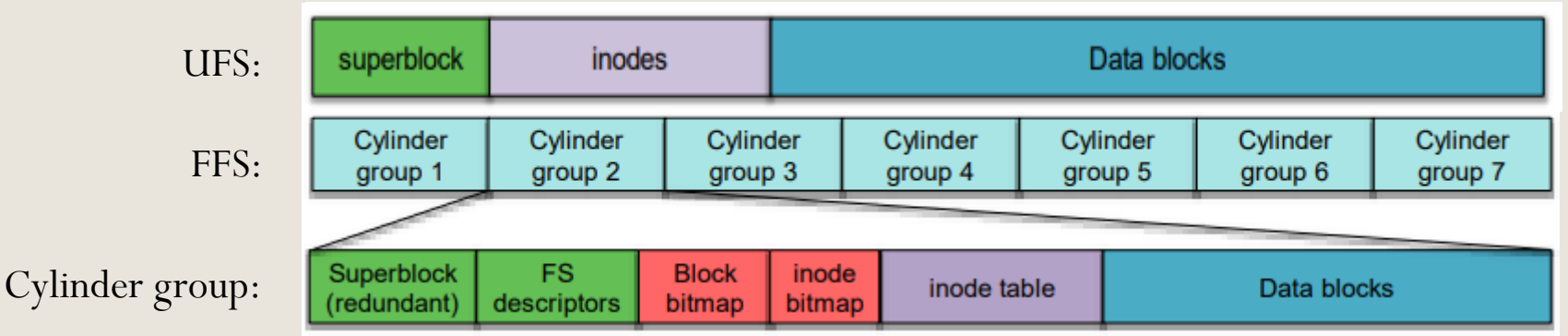
- Each inode contains direct, indirect, double-indirect, and triple-indirect blocks.
- A disk is divided into 3 parts: superblock, inodes, and data blocks.




- Superblock contains:
  - size of file system, number of free blocks, list of free blocks, pointer to free block lists,
  - index of the next free block in the free block list, size of the inode list,
  - number of free inodes in the file system, index of the next free inode in the free inode list,
  - modified flag (clean/dirty)
- Free space is managed as a linked list of free blocks.
  - Eventually every disk block access will require a seek.

# BSD Fast File System (FFS)

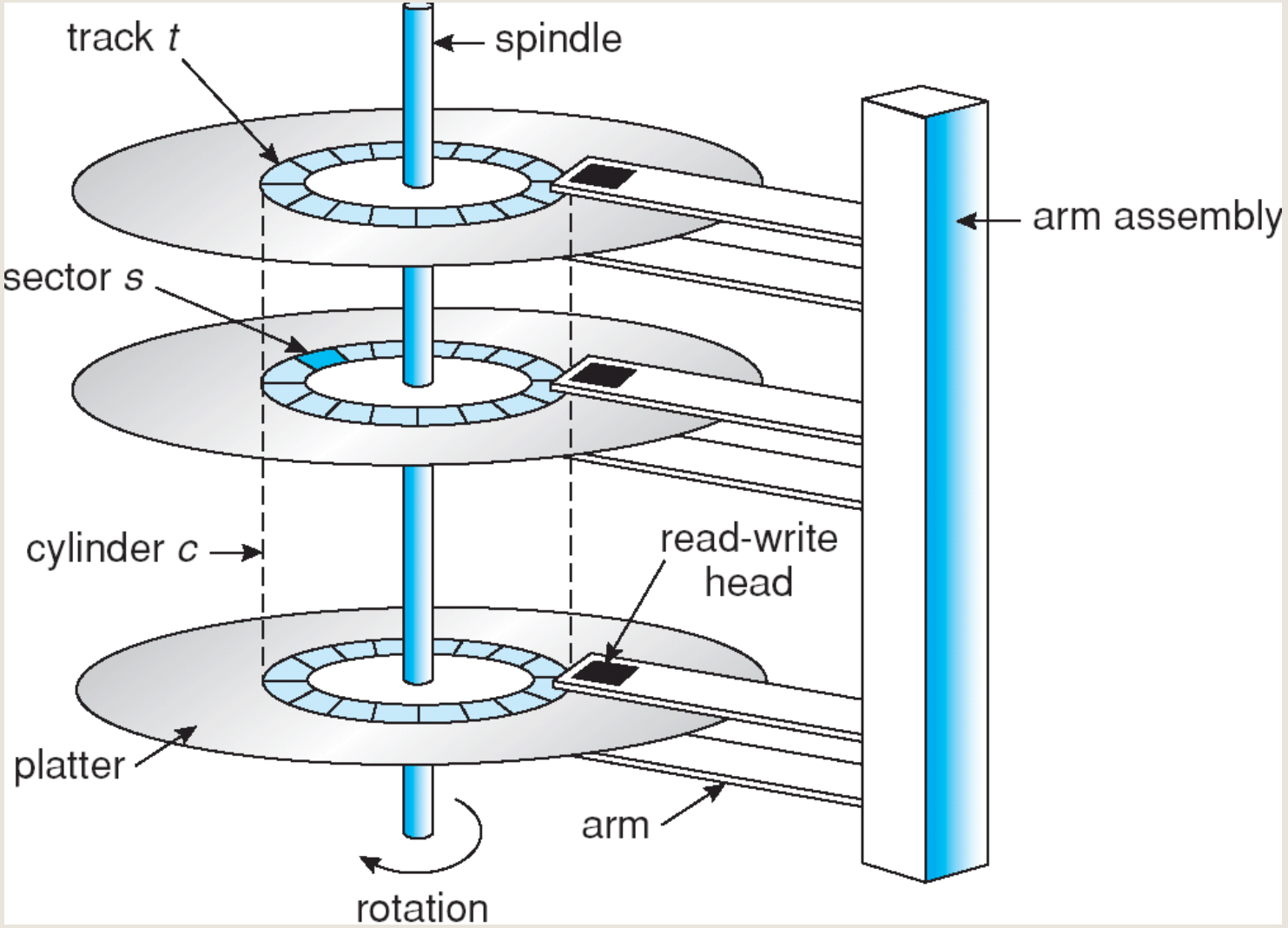
- It tries to minimize head movement by reduce seek time.
  - It keeps file data blocks close to its inode to minimize seek time to fetch data.
  - It also keeps related files and their directories together.
  - A cylinder is a collection of all blocks on the same disk track on all heads of a disk.
  - A cylinder group is a collection of blocks on one or more consecutive cylinders



# Secondary Storage System

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - Transfer rate is the rate at which data flow between drive and computer
  - Positioning time (random-access time) is
    - time to move disk arm to desired cylinder (seek time) and
    - time for desired sector to rotate under the disk head (rotational latency)
  - Head crash results from disk head making contact with the disk surface
- Disks can be removable
- Drive attached to computer via I/O bus
  - The usual bus systems are USB, Fiber Channel, SCSI, etc.
  - Host controller in computer uses bus to talk to disk controller.

# Moving-Head Disk Mechanism

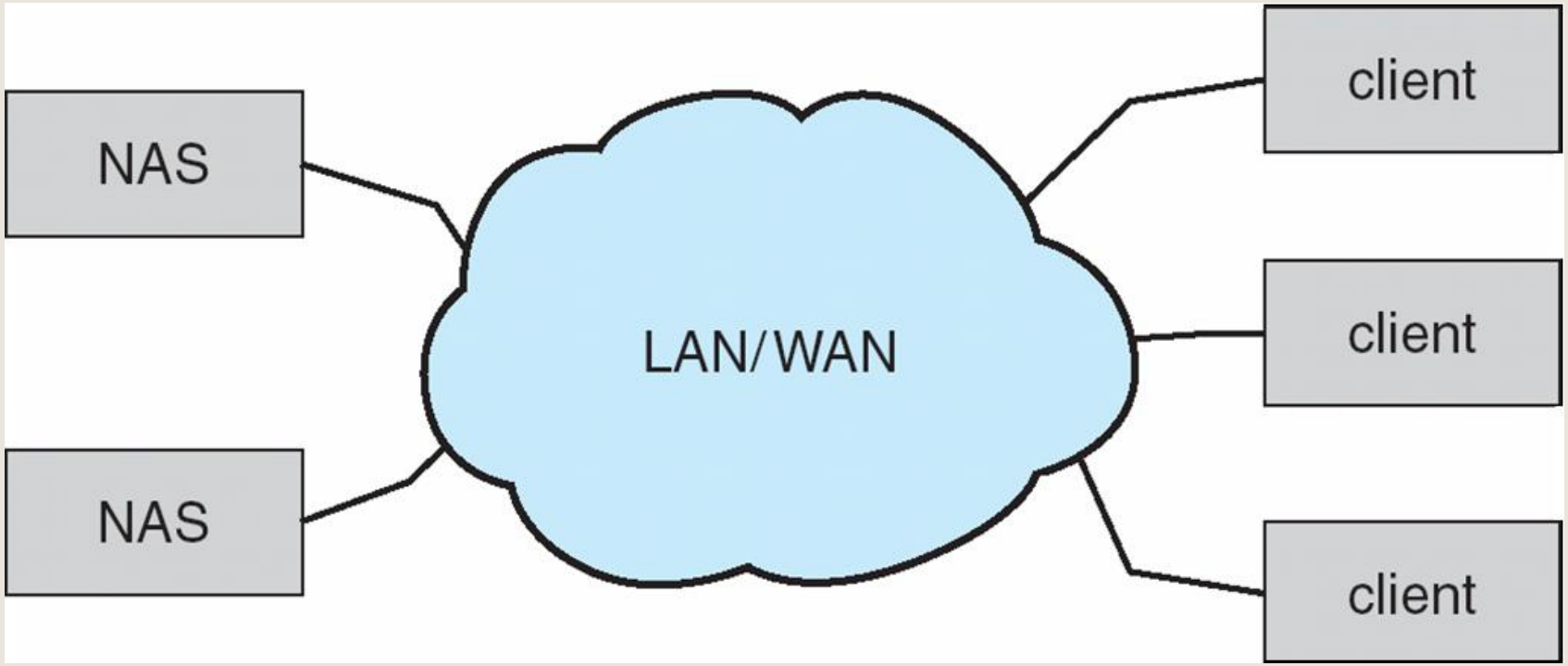


# Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer.
  - The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
    - Sector 0 is the first sector of the first track on the outermost cylinder
    - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
- Disks can be attached to a computer in the following ways.
  - Host-attached storage accessed through I/O ports talking to I/O busses.
    - SCSI is a commonly used I/O bus that can attach up to 16 devices on one cable.
      - SCSI initiator requests operation and SCSI targets perform tasks. Each target can have up to 8 disks.
  - Network-attached storage (NAS) is storage made available over a LAN rather than a local bus.
- Storage Area Network (SAN) is common in large storage environments.

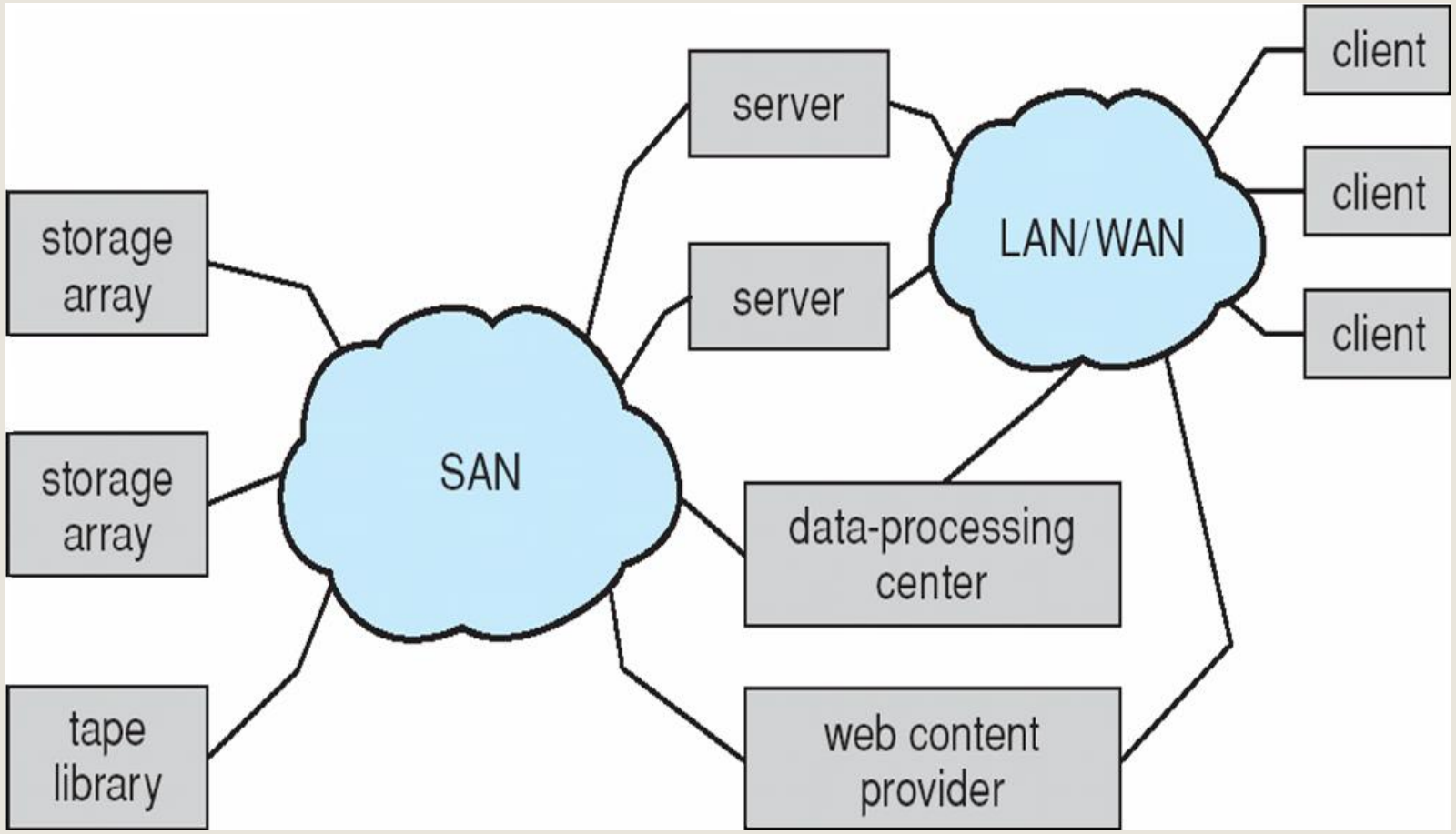
# Network-Attached Storage (NAS)

- NFS and CIFS are common protocols that were implemented via remote procedure calls (RPCs) between host and storage.
- The iSCSI protocol that uses IP network to carry the SCSI protocol is also common,



# Storage Area Network (SAN)

- Multiple hosts can be attached to multiple storage arrays.



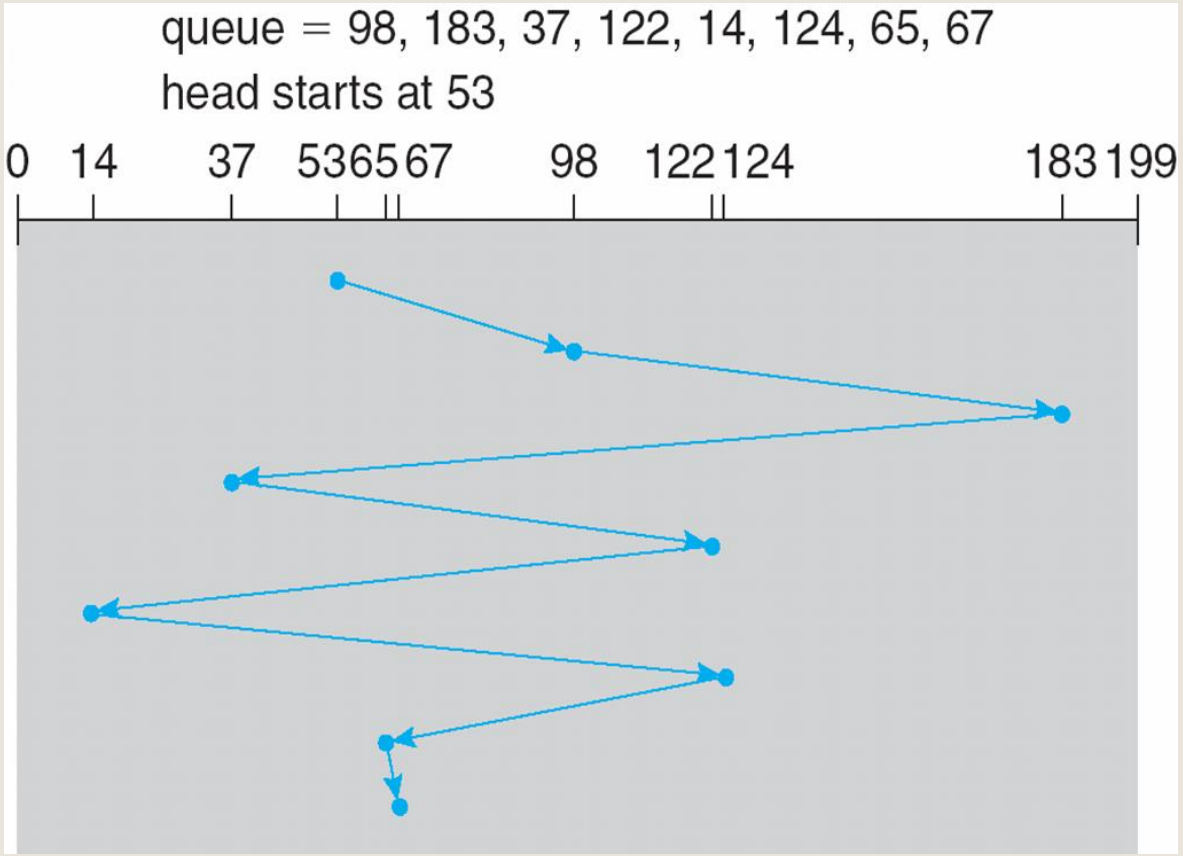
# Disk Scheduling

- The OS is responsible for using hardware efficiently
  - For the disk drives, this means having a fast access time and high disk bandwidth
- Access time has two major components
  - Seek time is the time for the disk to move the heads to the cylinder containing the desired sector
  - Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head. In average, it is one half of the rotation time of the disk.
  - The seek time is far larger that the rotational latency ➔ Minimize seek time
- Seek time ≈ seek distance (the distance between current and new head positions)
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

# Disk Scheduling Algorithms

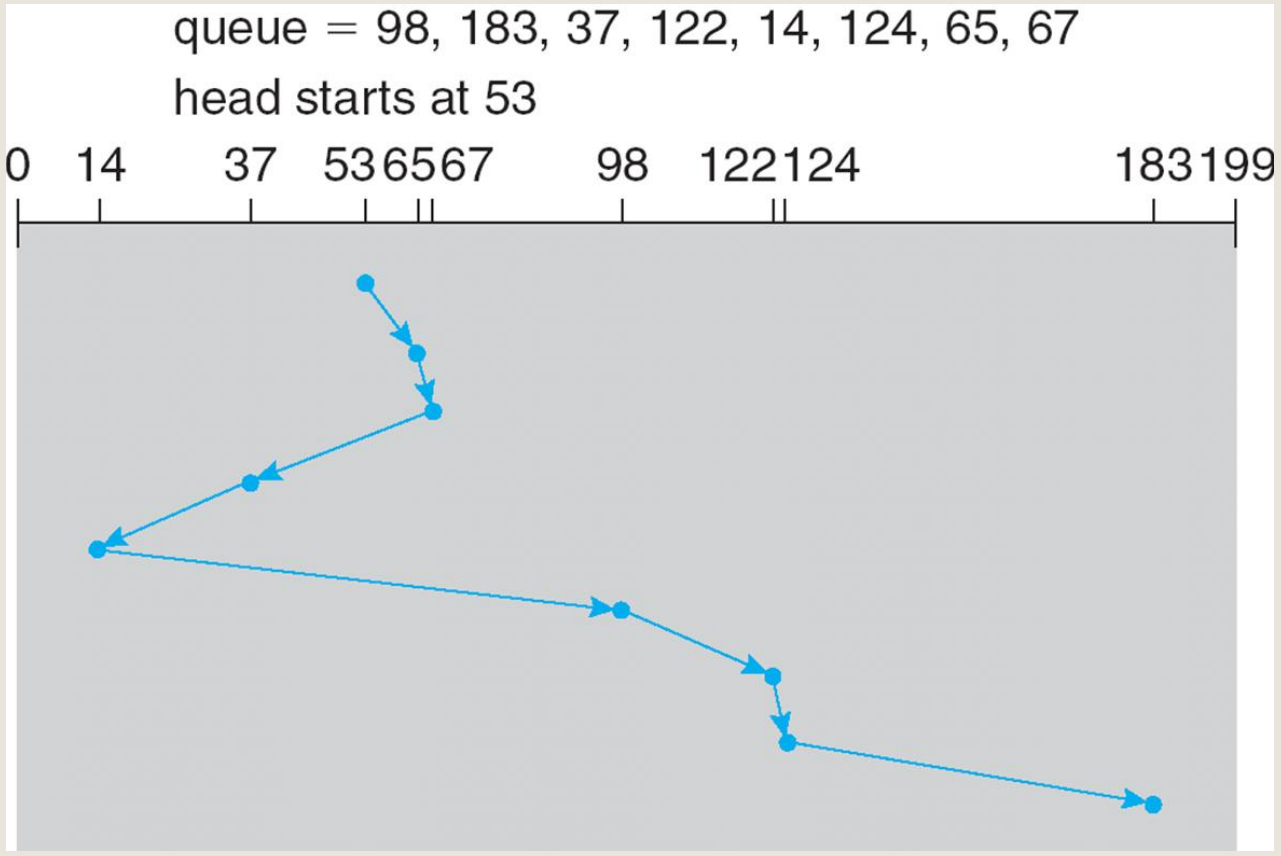
- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate them with a request sequence (cylinder numbers: 0-199) of 98, 183, 37, 122, 14, 124, 65, 67, where the Current head position is 53.
- The FCFS handles requests according to their arriving order.
- The SSTF (shortest seek time first) selects the request with the minimum seek time from the current head position.
- The SCAN schedules pending I/O in the sequence of the current head moving direction. When the head reaches the end, it reverses the moving direction.
  - In the LOOK schedule, the head only goes as far as the last request in each direction.
- The C-SCAN (Circular SCAN) schedules pending I/O in the sequence of the head moving direction. When the head reaches the end, it fast moves the head back to the other end. It serves the requests only in one direction.

# FCFS




The total seek time is (98-53) + (183-98) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65) = 45+85+146+85+108+110+57+2= 638

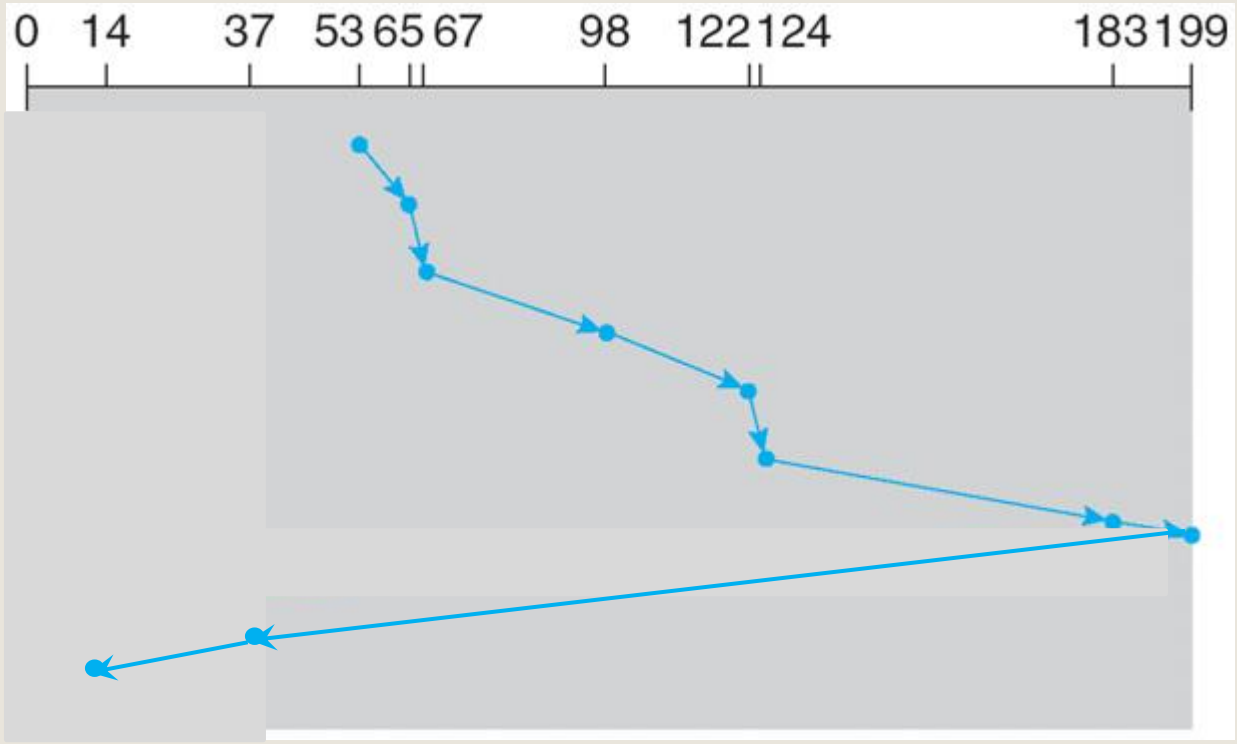
# SSTF




The total seek time is (65-53) + (67-65) + (67-37) + (37-14) + (98-14) + (122-14) + (124-122) + (183-124) = 12+2+30+23+84+108+2+59= 320

# SCAN

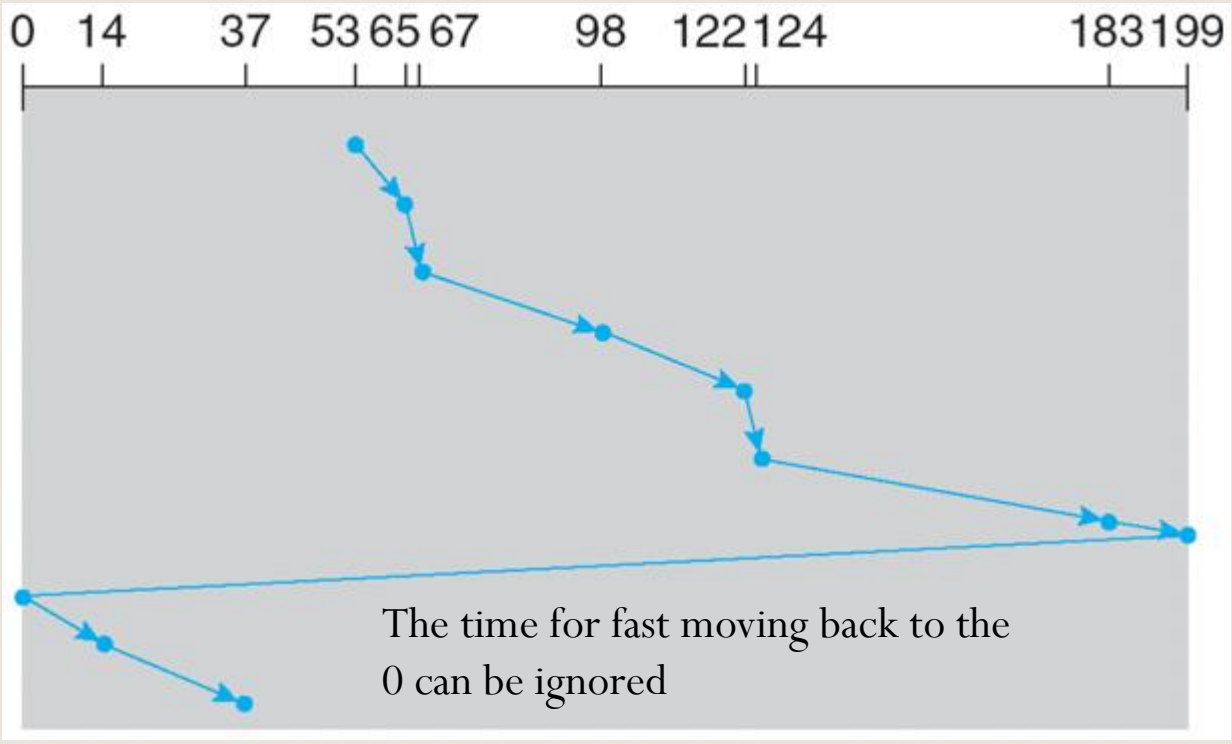
Queue = 98, 183, 37, 122, 14, 124, 65. 67
Head is at 53 and is moving from 0 to 199




The total seek time is (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-37) + (37-14) = (199-53) + (199-14) = 146+185 = 331

# C-SCAN

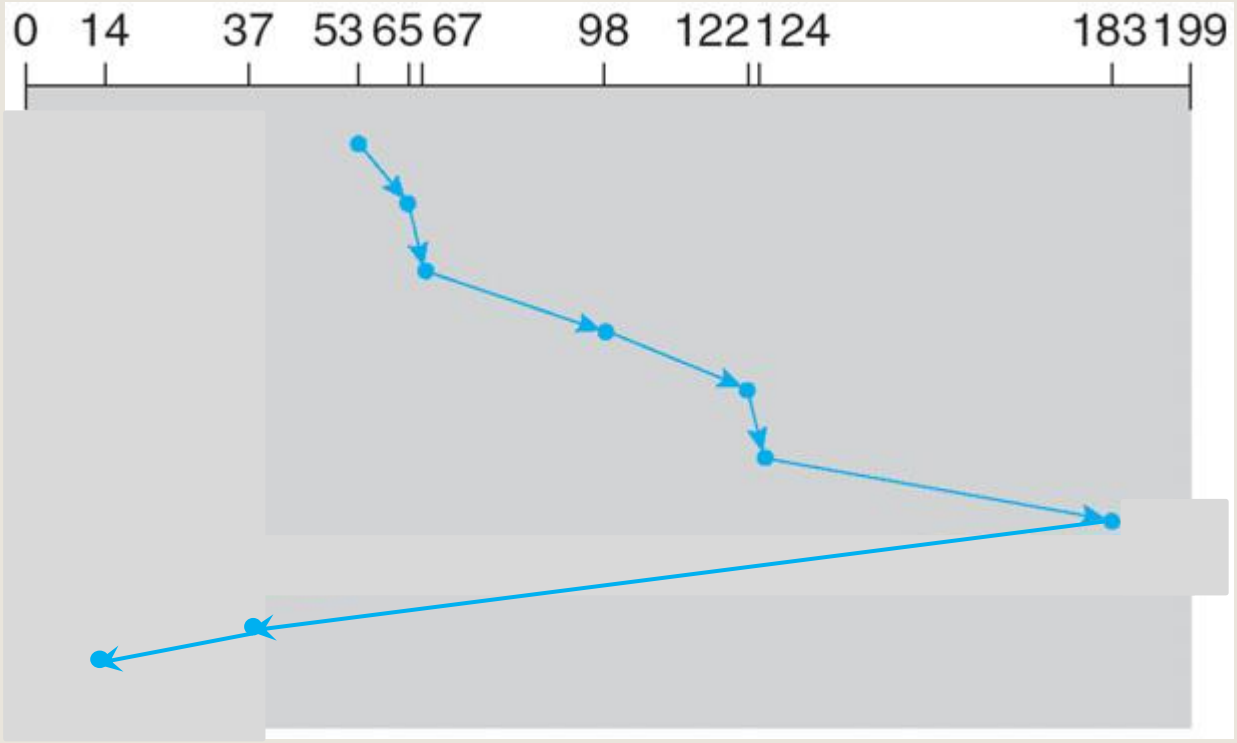
Queue = 98, 183, 37, 122, 14, 124, 65. 67
Head is at 53 and is moving from 0 to 199




The total seek time is (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) ~~+ (199→0)~~ + (14-0) + (37-14) = (199-53) + (37-0) = 146+37 = 183 + ??

# LOOK

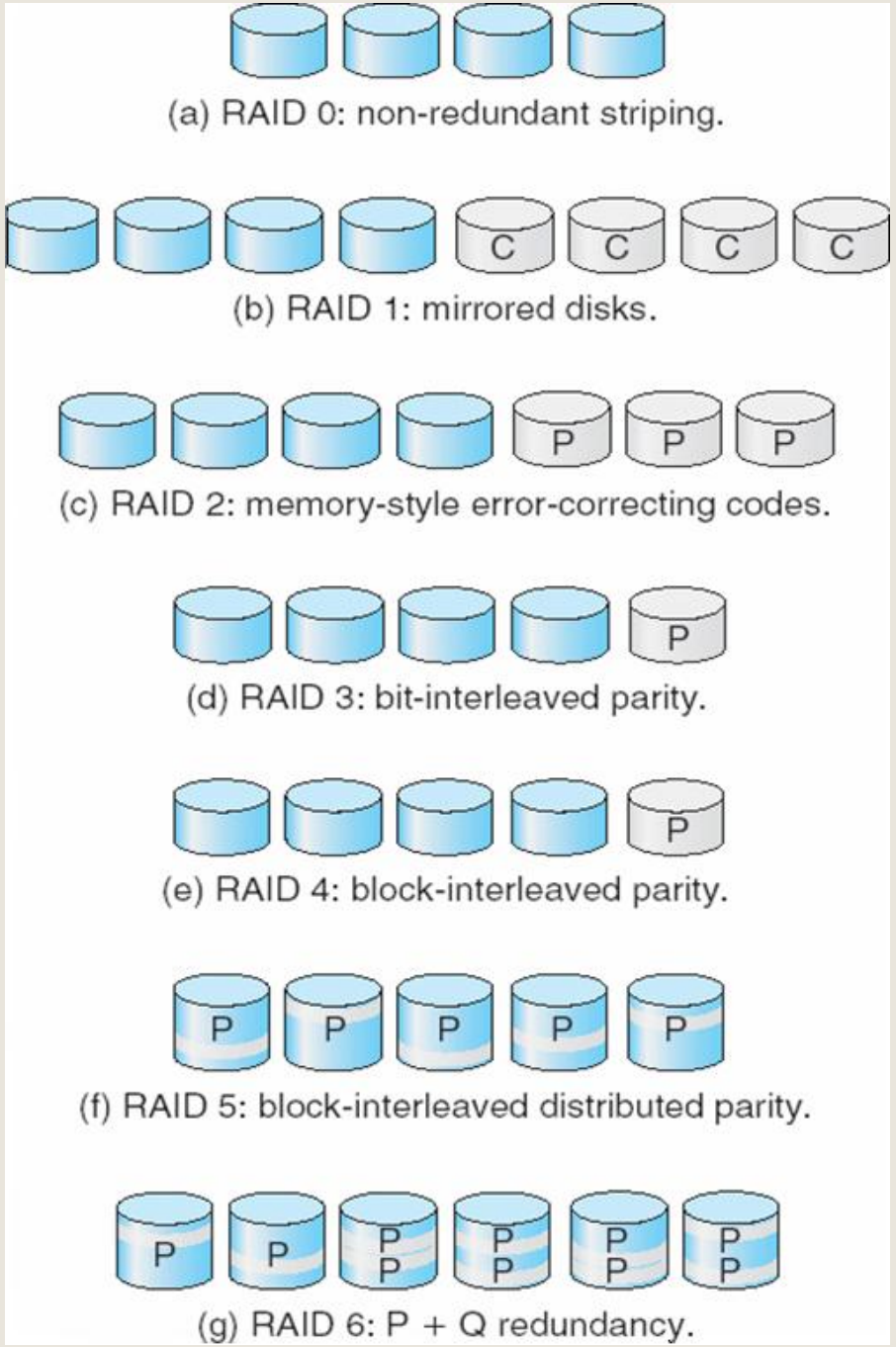
Queue = 98, 183, 37, 122, 14, 124, 65. 67
Head is at 53 and is moving from 0 to 199

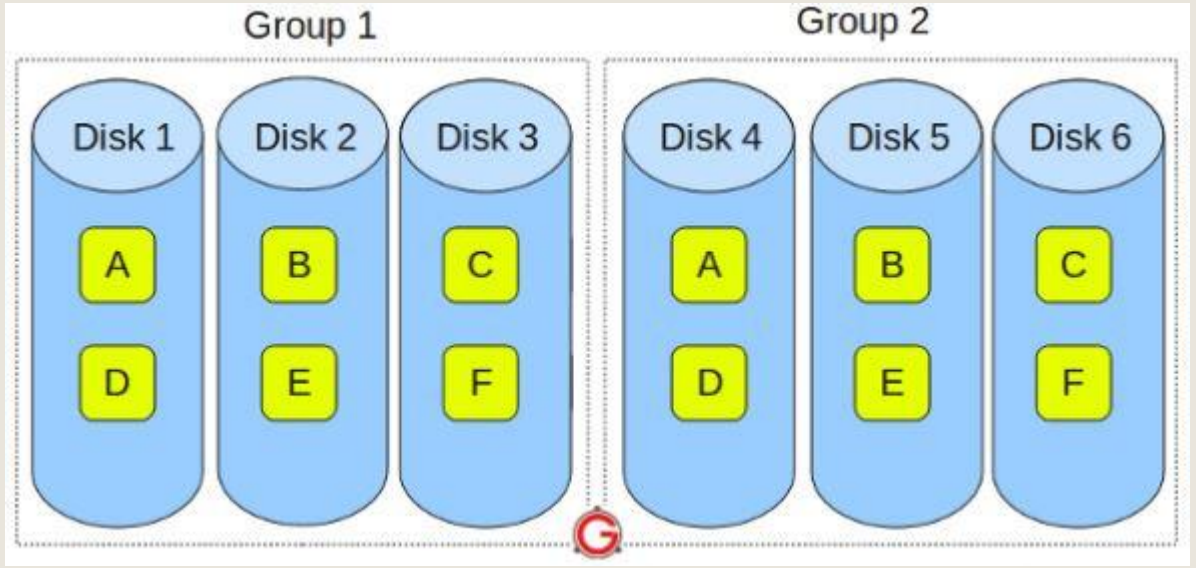



The total seek time is (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-37) + (37-14) = (183-53) + (183-14) = 130+169 = 299

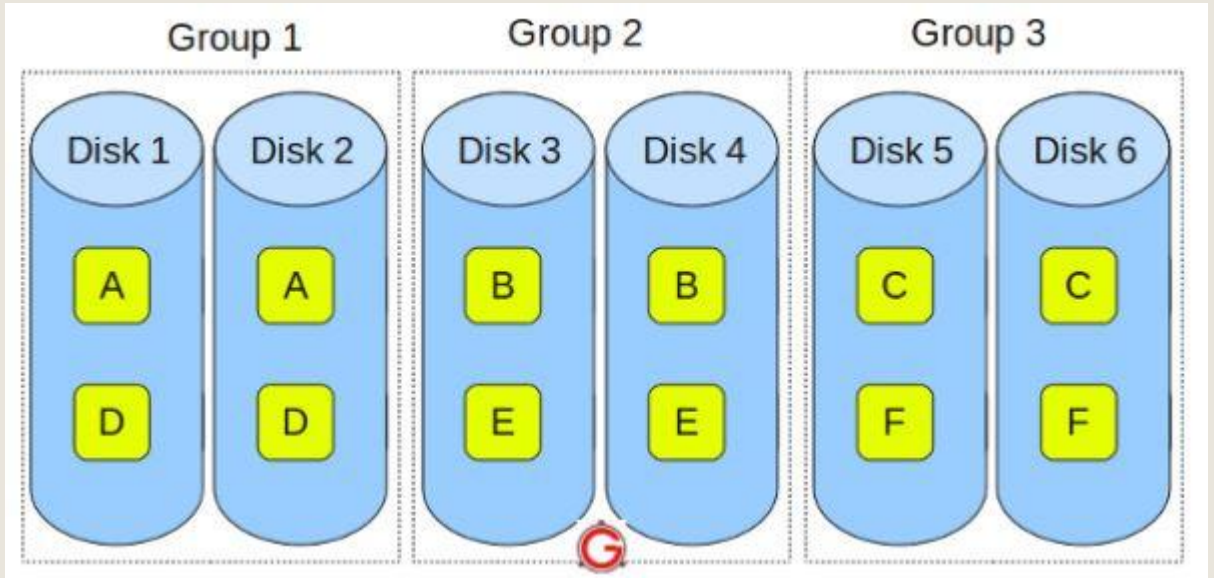
# RAID Structure

- RAID stands for Redundant Array of Independent (or Inexpensive) Disks
  - It contains multiple disk drives to provide reliability via redundancy
  - It can increases the mean time to failure
- There are 7 different levels of RAID: RAID-0 to RAID-6
  - RAID 0 uses disk striping to group disks into one storage unit.
  - Mirroring or shadowing (RAID 1) keeps duplicate of each disk.
    - Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provides high performance and high reliability
  - Block interleaved parity (RAID 4, 5, 6) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic replication of the data between arrays is common
- Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them




Raid 0+1




Raid 1+0

# Raid 1+0 vs Raid 0+1

- RAID 1+0 is also called as "stripe of mirrors" has better fault tolerance level. (why?)
  - It groups the disks in pair of two (for mirror.)
  - For 6 disks, there will be three groups : Group 1, Group 2, and Group 3.
    - Within each group, the data is mirrored. Thus, block A written on Disk 1 will be mirrored on Disk 2. Block B written on Disk 3 will be mirrored on Disk 4.
    - Across the group, the data is striped. Block A is written to Group 1, Block B is written to Group 2, Block C is written to Group 3.
- RAID 0+1 is also called as "mirror of stripes"
  - It groups the disks into two groups (for mirror.)
  - For 6 disks, it create two groups with 3 disks each.
    - Within each group, the data is striped. Thus, block A is written to Disk 1, block B to Disk 2, block C to Disk 3.
    - Across the group, the data is mirrored. The Group 1 and Group 2 will look exactly the same.