OS Review 3

Chapter 10 - 13

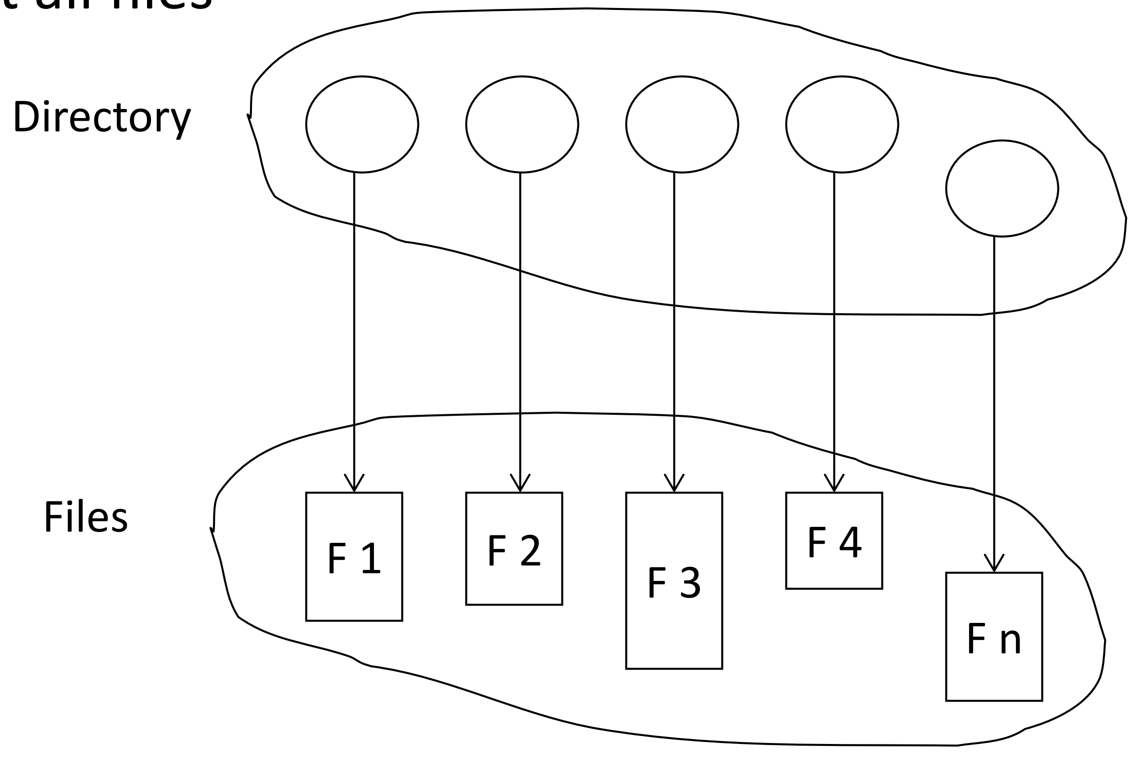
Chapter 10: basic of file

- What is a file?
  - Contiguous logical address space
  - A sequence of bits, bytes, lines, or records. The meaning is defined by the creator and user.

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Directory Structure

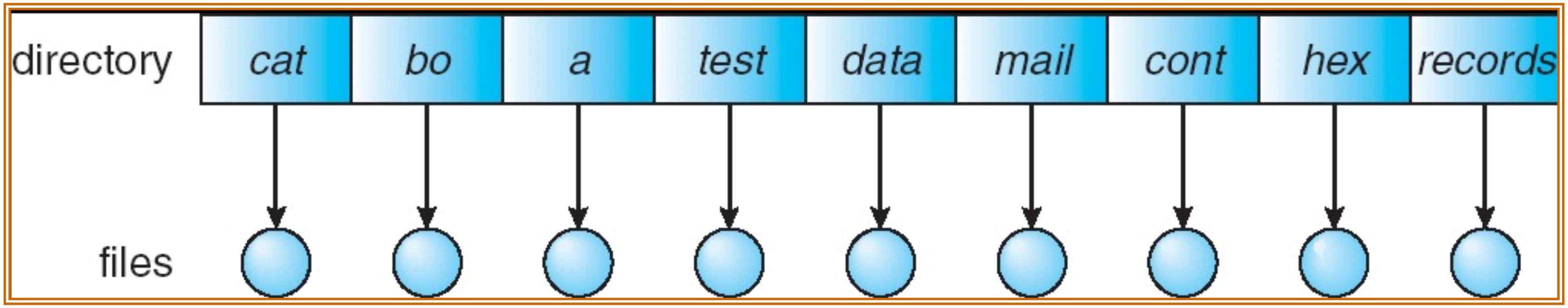
- A collection of nodes containing (management) information about all files



Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

Single-Level Directory

- A single directory for all users

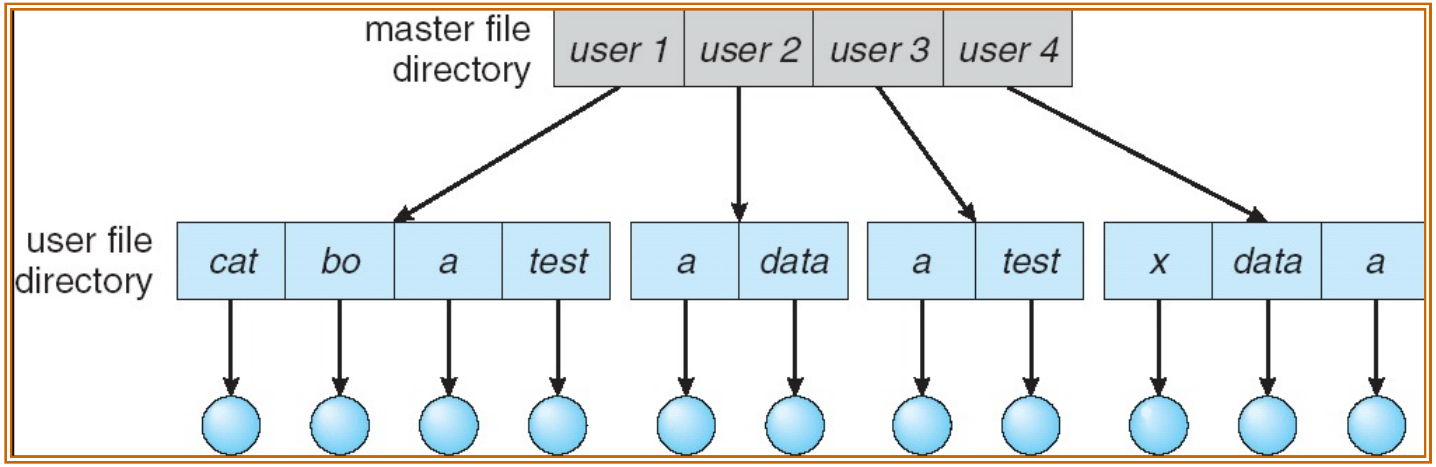


Naming problem

Grouping problem

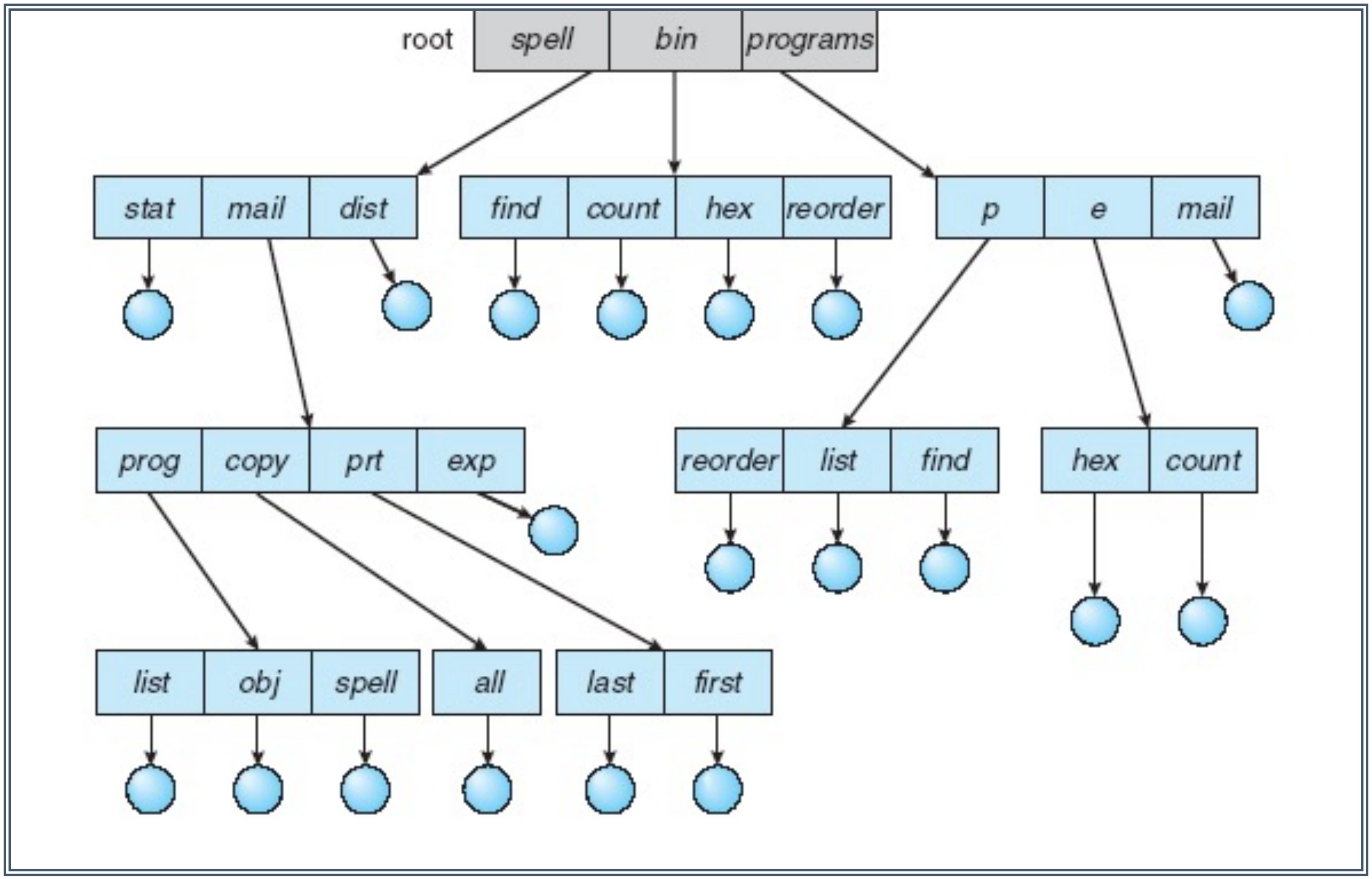
Two-Level Directory

- Separate directory for each user



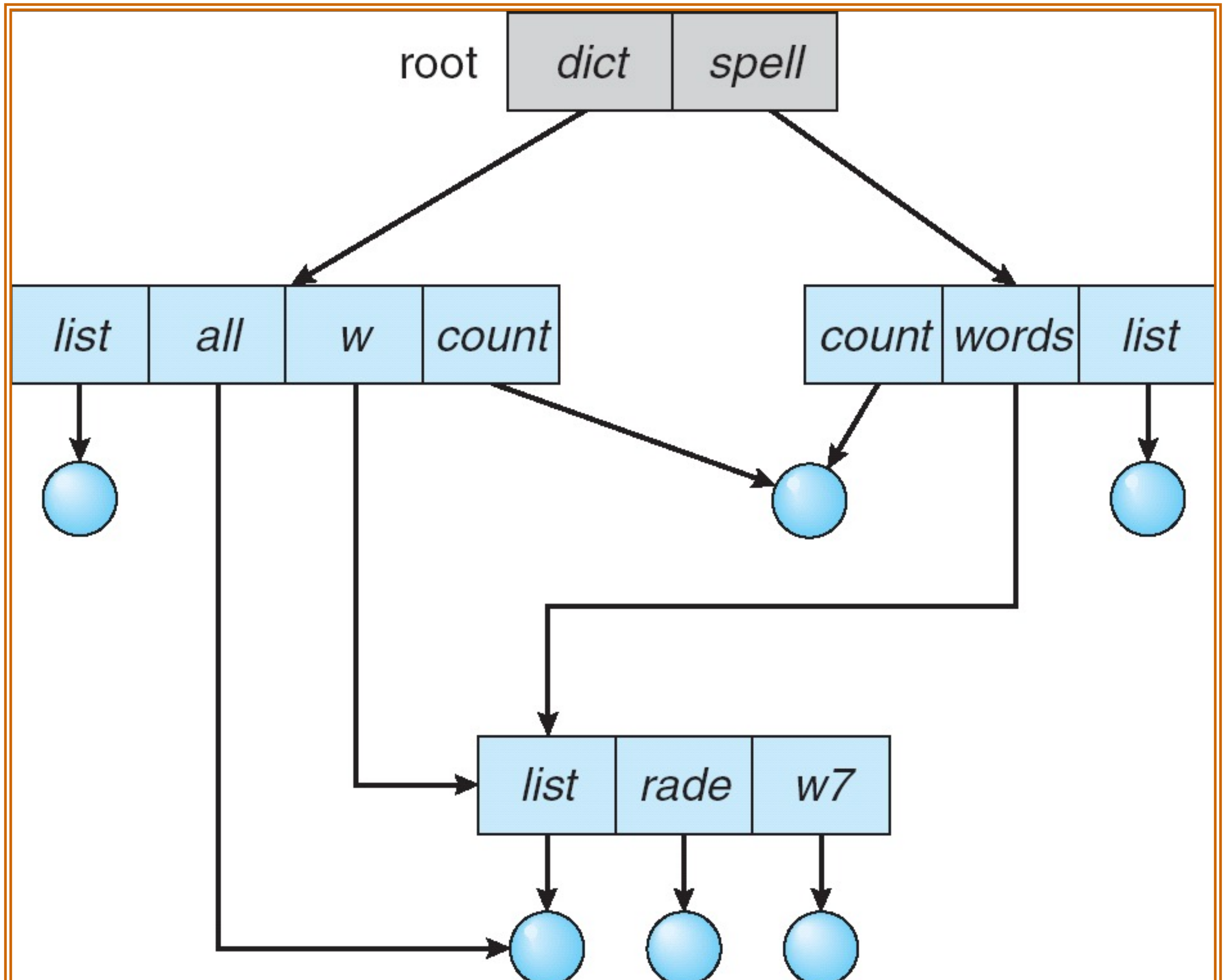
- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories

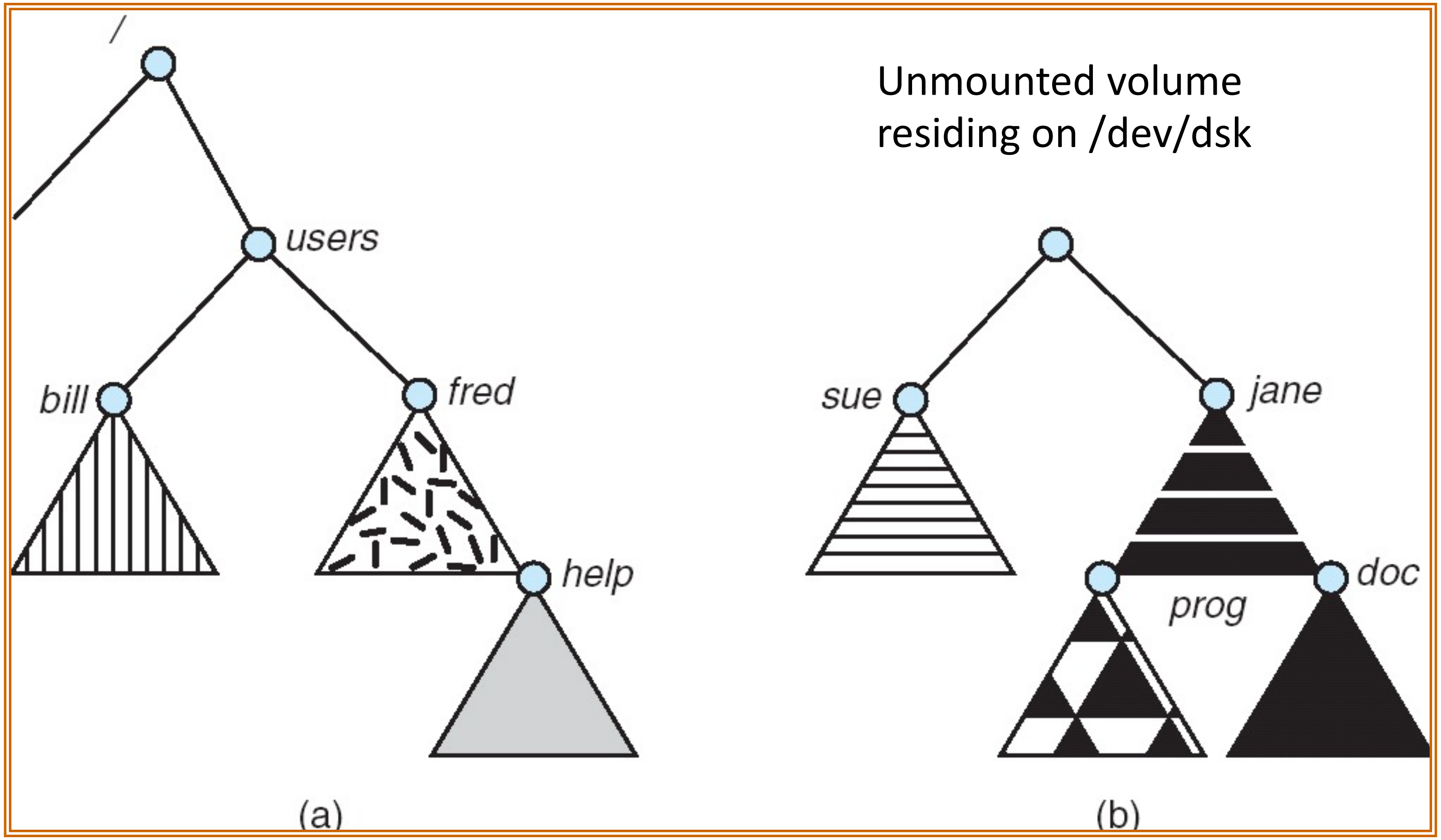


Acyclic-Graph Directories

- Requirement for file sharing
- Have shared subdirectories and files



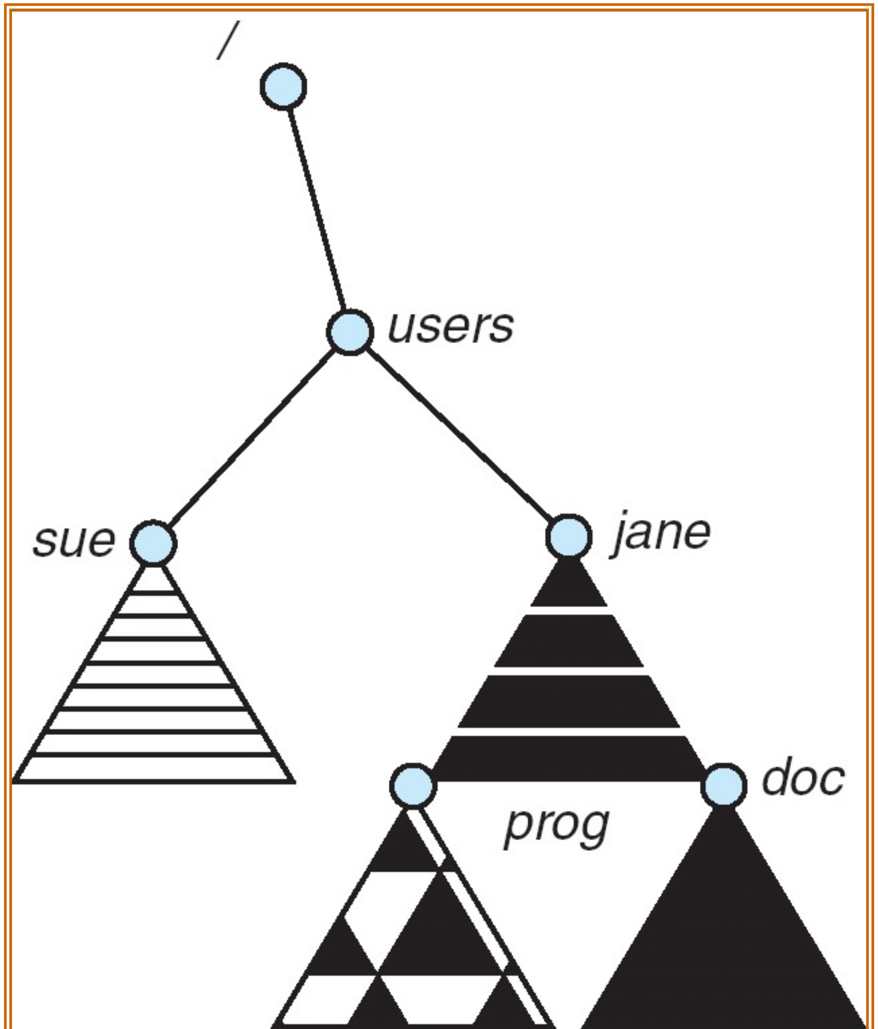
Mount



/
users
bill
fred
help
Unmounted volume
residing on /dev/dsk
sue
jane
prog
doc
(a)
(b)

Mount Point

```
$ mount  /dev/dsk  /users
```



Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) **owner access**	7	$\Rightarrow$	1 1 1
			RWX
b) **group access**	6	$\Rightarrow$	1 1 0
			RWX
c) **public access**	1	$\Rightarrow$	0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.

owner group public

chmod 761 game

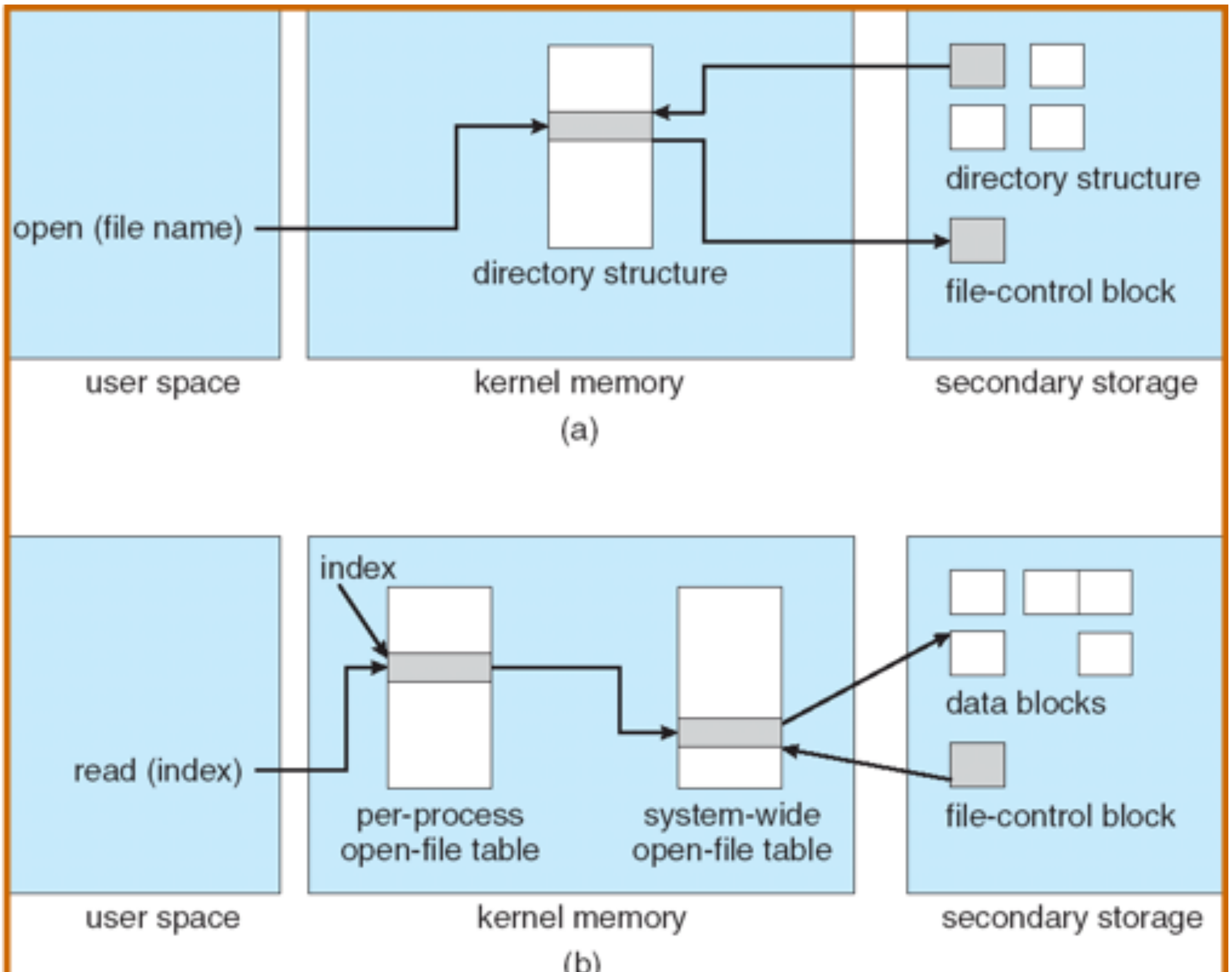
Attach a group to a file

chgrp G game

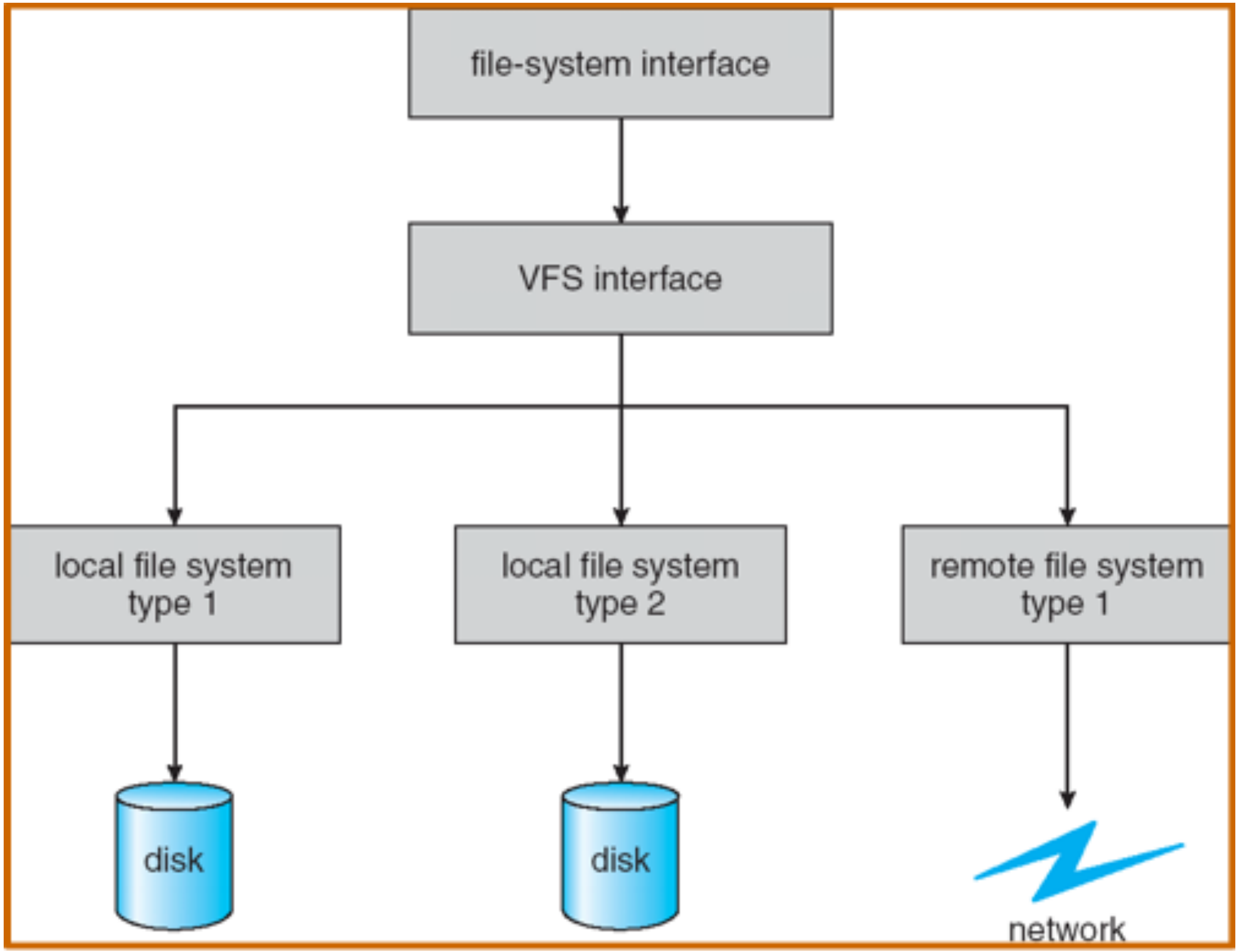
Chapter 11: File System Implementation

- Data structures
  - Disk structures
    - Boot control block
    - Volume control block
    - Directory structure per file system
    - Per-file FCB (inode in UFS, master file table entry in NTFS)
  - In-memory structures
    - In-memory mount table about each mounted volume
    - Directory cache
    - System-wide open-file table
    - Per-process open-file table

In-Memory File System Structures



Semantic of Virtual File System



Allocation Methods

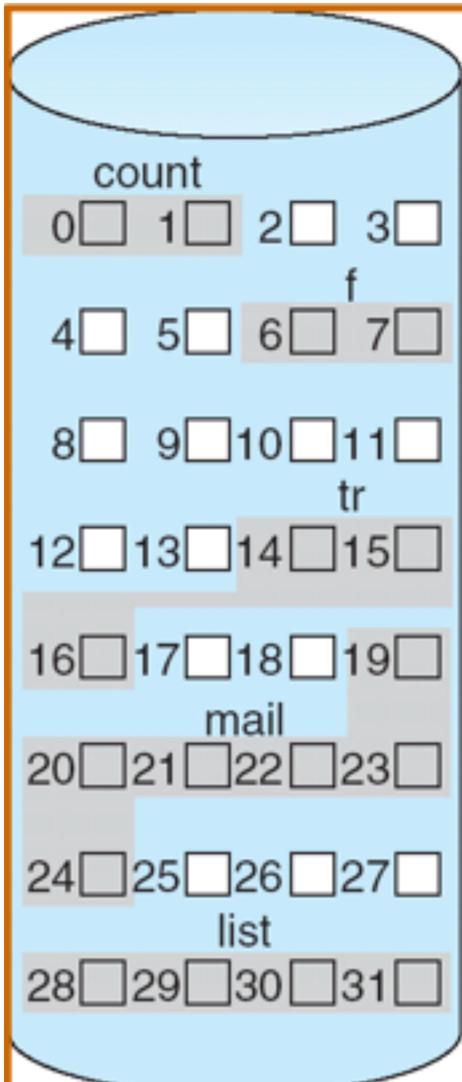
- An allocation method refers to how disk blocks are allocated for files:

- **Contiguous allocation**

- **Linked allocation**

- **Indexed allocation**

Contiguous Allocation



directory

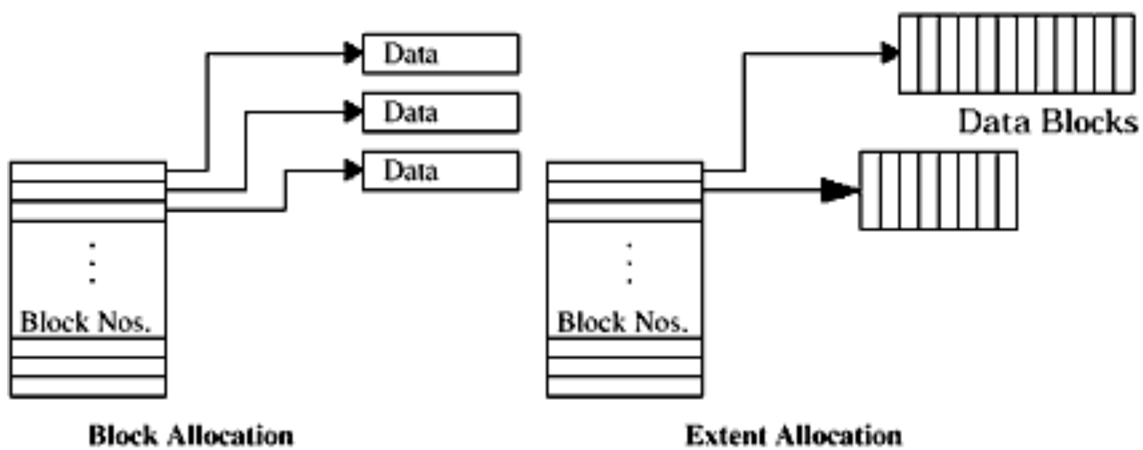
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

LA/512 → Q, R

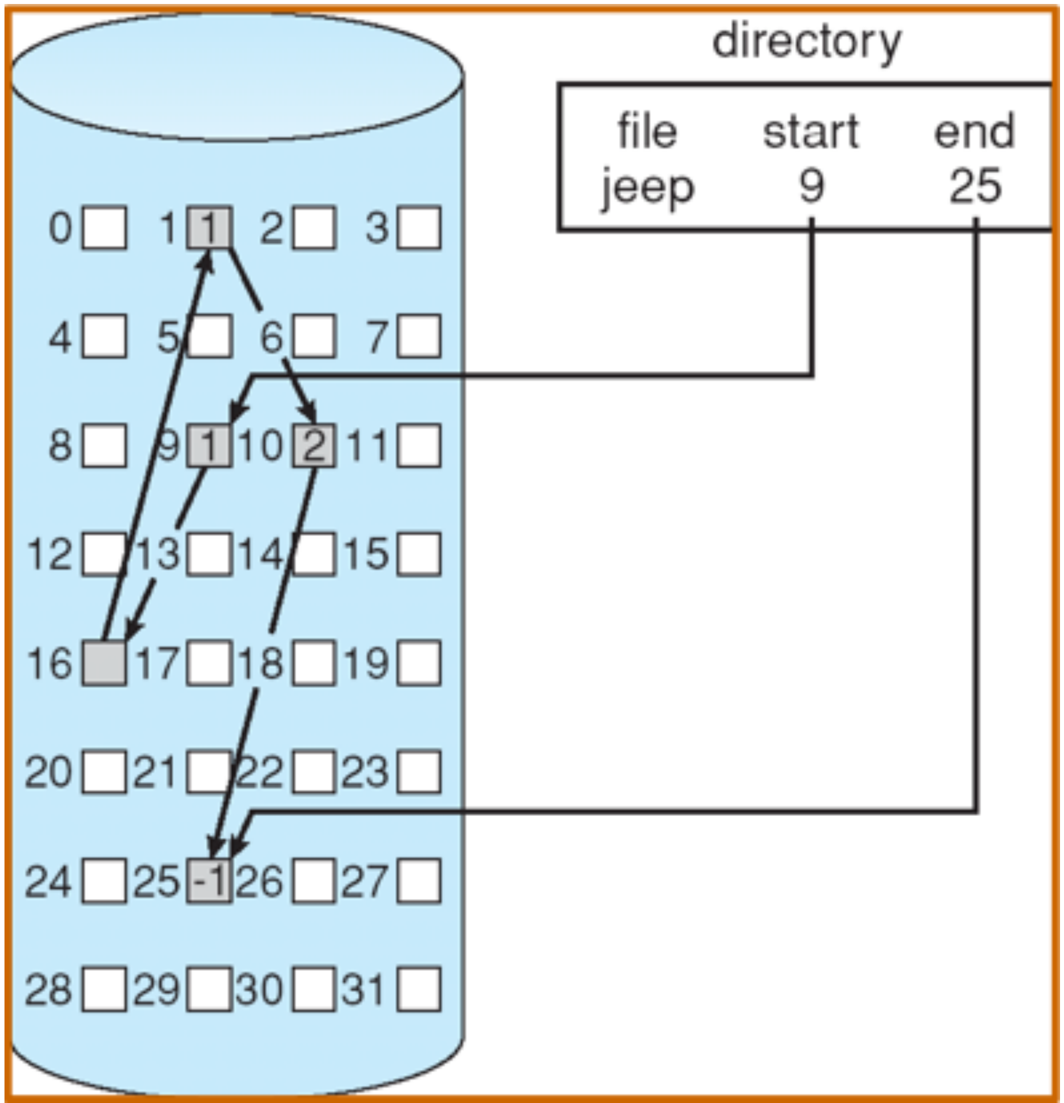
Block to be accessed = Q + start_address
Displacement into block = R

Extent-based Allocation

- Extent-based file systems allocate disk blocks in **extents**
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents.

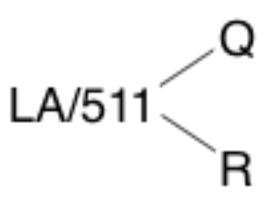


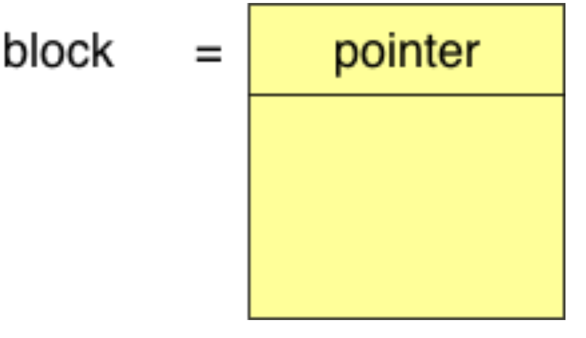
Linked Allocation



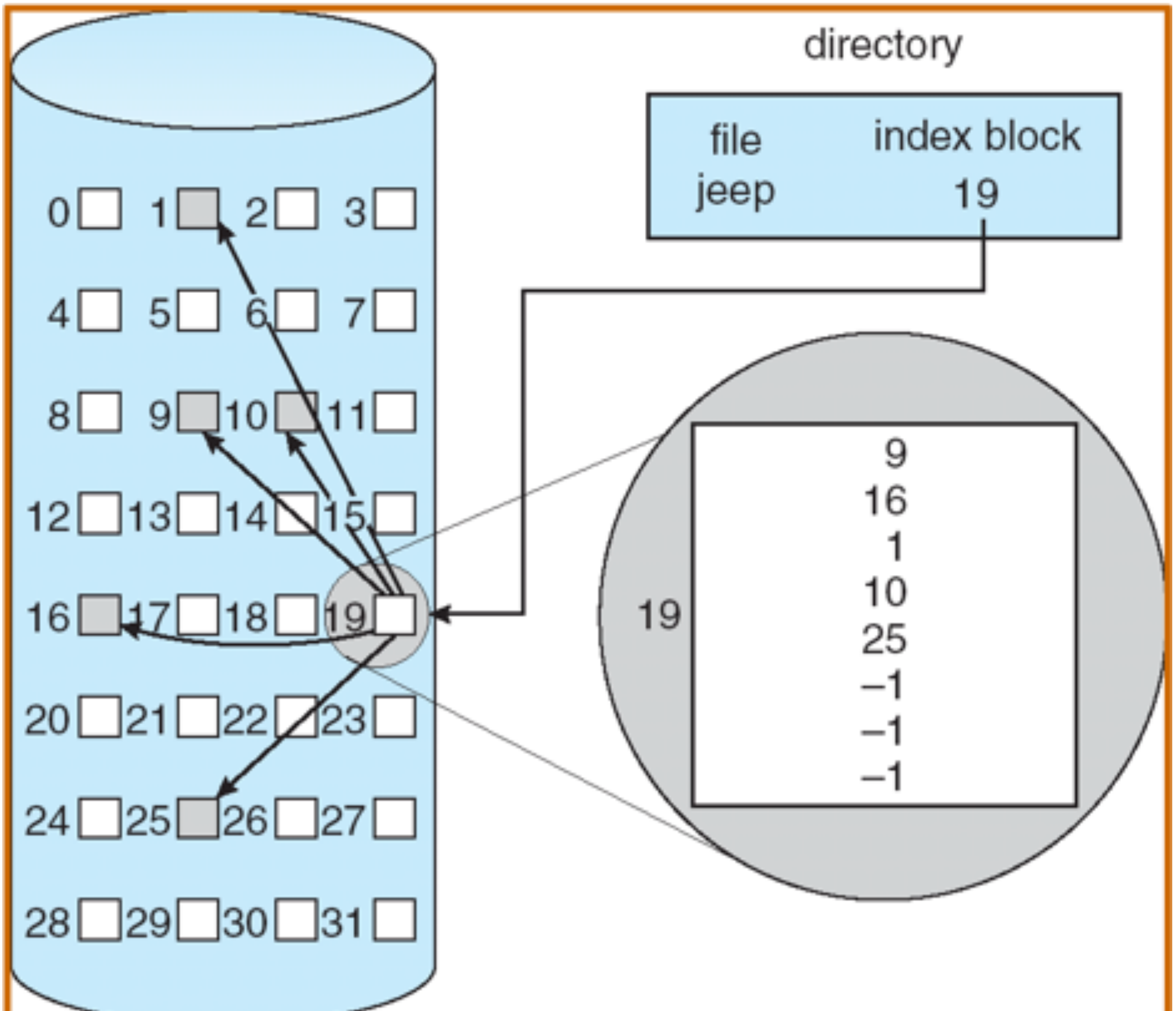
Linked Allocation

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access, poor reliability
- Mapping



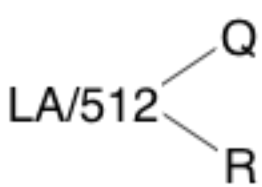


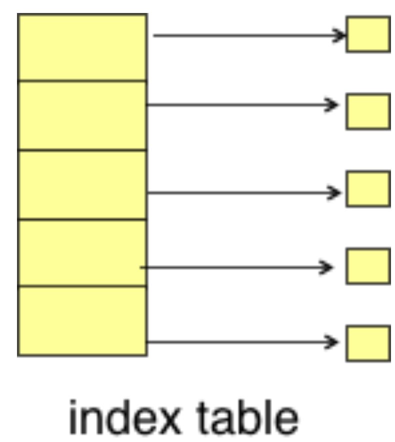
Indexed Allocation



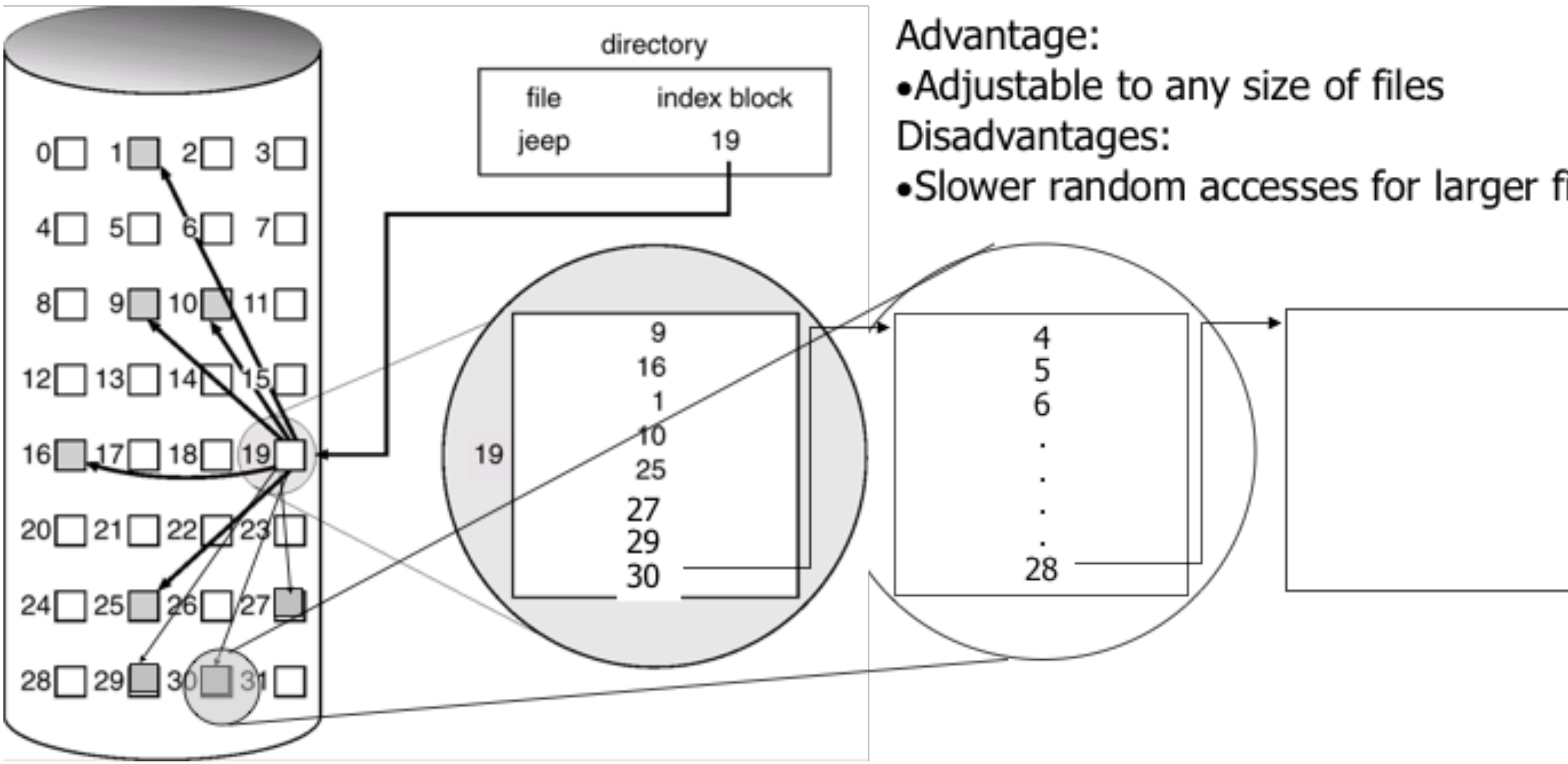
Indexed Allocation

- Need index table (analogous to page table)
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- When mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words (512=2^9, 2^9 * 2^9 = 2^18, 2^18 / 1024 = 256). We need only 1 block for index table.





Indexed Allocation



Advantage:

- Adjustable to any size of files

Disadvantages:

- Slower random accesses for larger files

Indexed Allocation

- When mapping from logical to physical in a file of unbounded length (block size of 512 words). – more pointers are needed
- Linked scheme – Link blocks of index table (no limit on size).

$$LA / (512 \times 511) \begin{cases} Q_1 \\ R_1 \end{cases}$$

Q_1 = block of index table

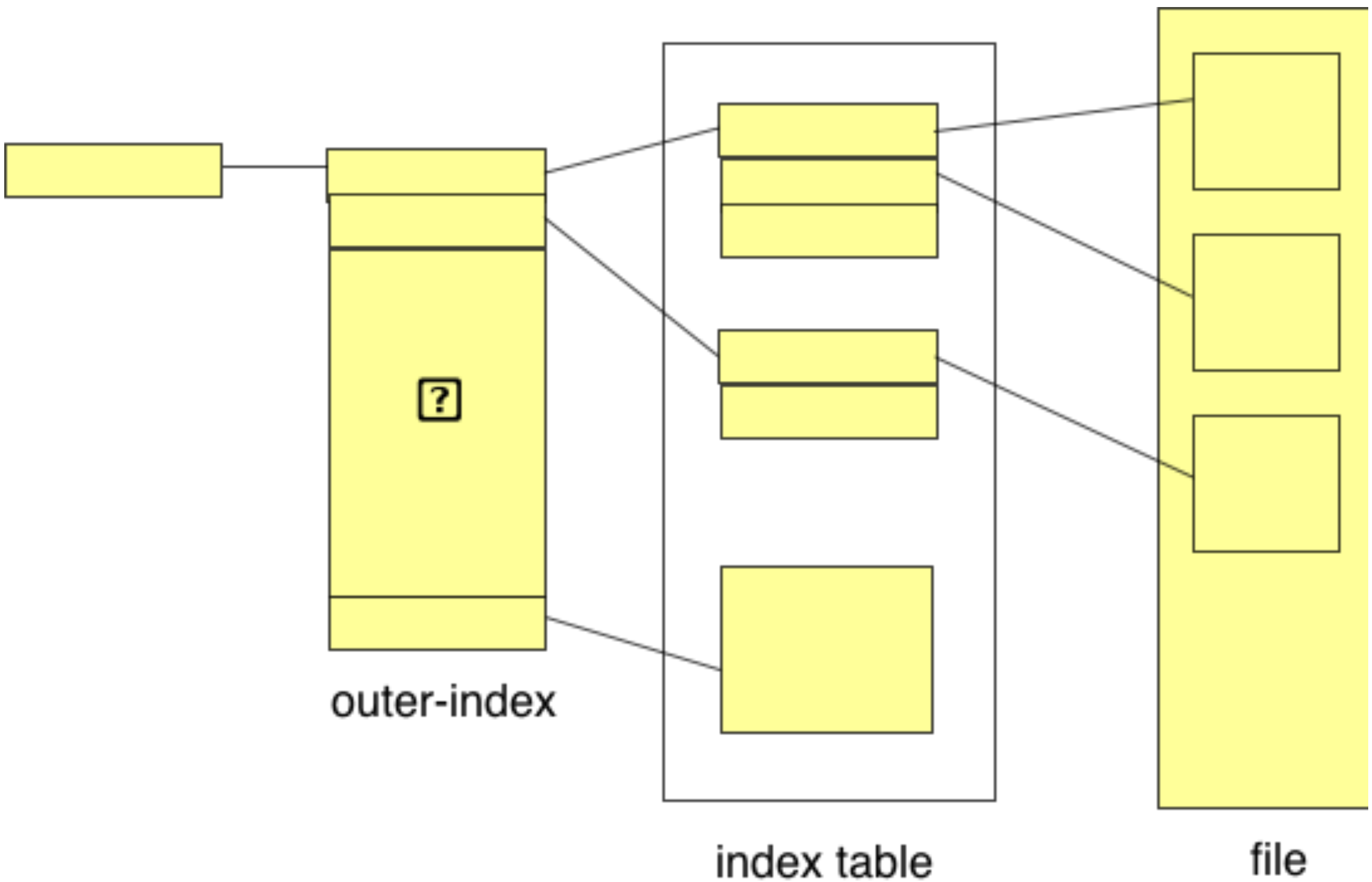
R_1 is used as follows:

$$R_1 / 512 \begin{cases} Q_2 \\ R_2 \end{cases}$$

Q_2 = displacement into block of index table

R_2 displacement into block of file:

Two-level Indexed Allocation



Two-level Indexed Allocation

- Two-level index (maximum file size is 512^3)

$$LA / (512 \times 512) \left\langle \begin{matrix} Q_1 \\ R_1 \end{matrix} \right.$$

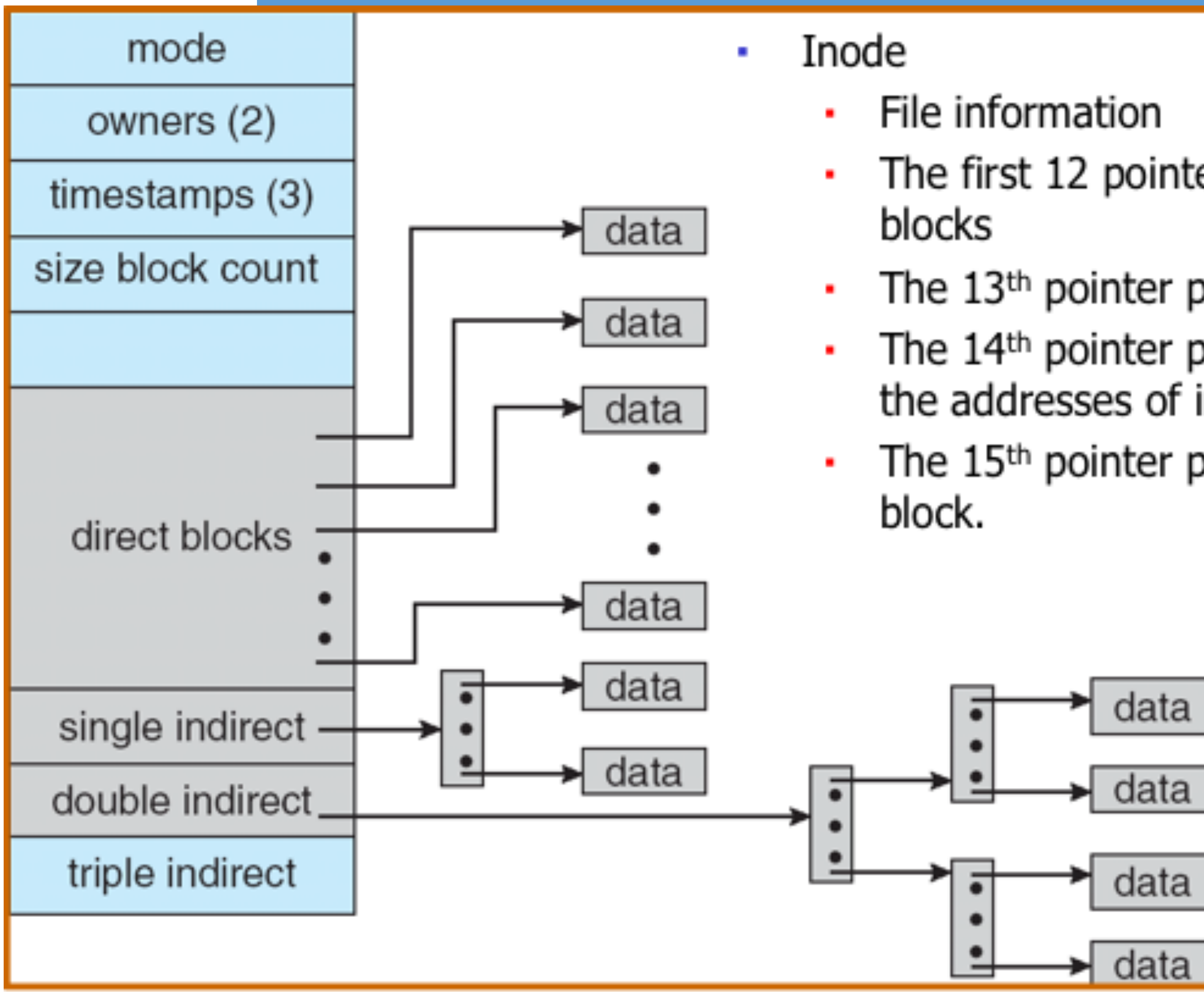
Q_1 = displacement into outer-index
R_1 is used as follows:

$$R_1 / 512 \left\langle \begin{matrix} Q_2 \\ R_2 \end{matrix} \right.$$

Q_2 = displacement into block of index table
R_2 displacement into block of file:

Unix Disk Allocation

$4K*(12+4K/4 + 4K/4*4K/4+4K/4*4K/4*4K/4)>4T$



- Inode
  - File information
  - The first 12 pointers point directly to data blocks
  - The 13th pointer points to an index block
  - The 14th pointer points to a block containing the addresses of index blocks
  - The 15th pointer points to a triple index block.

Dentry and Inode

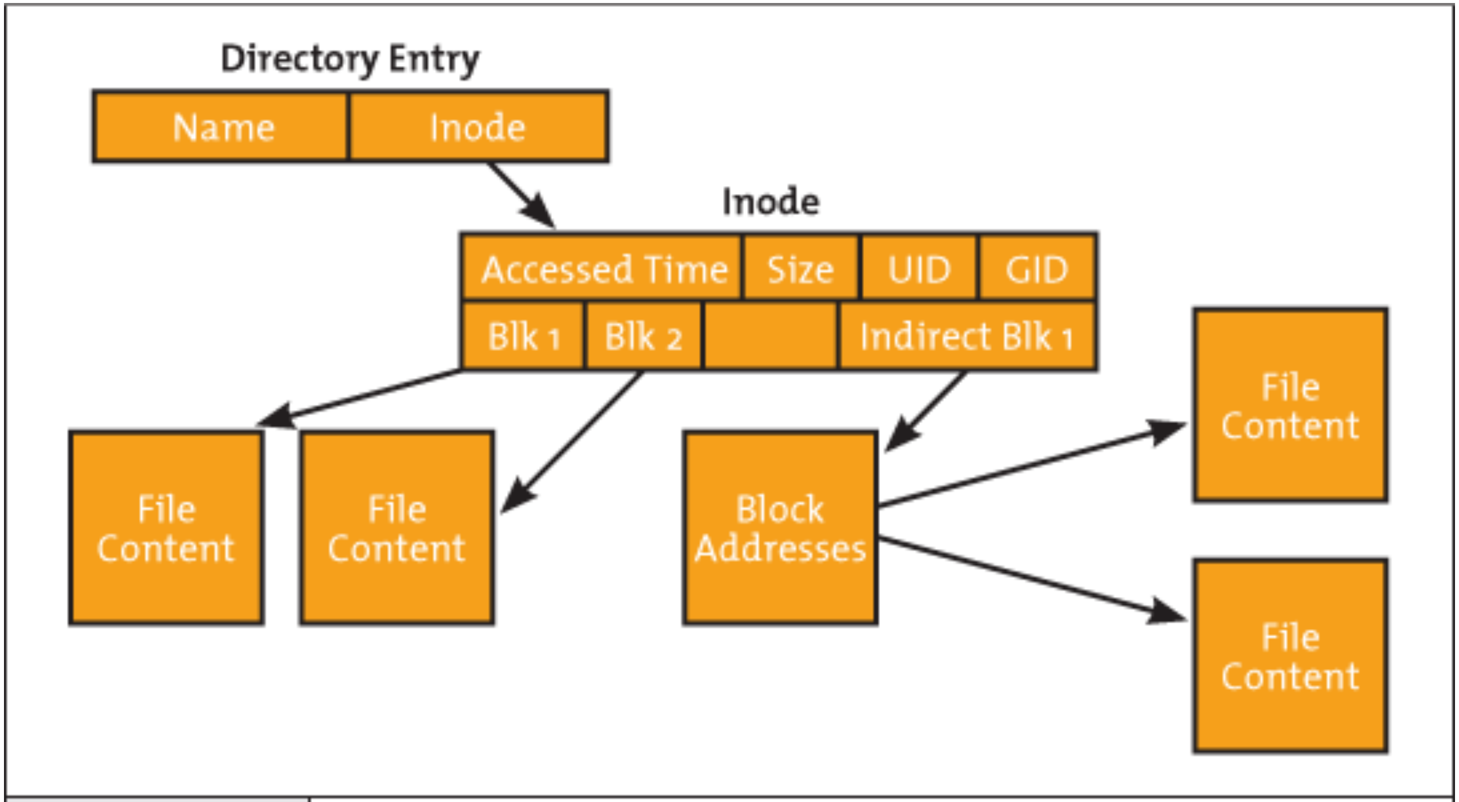
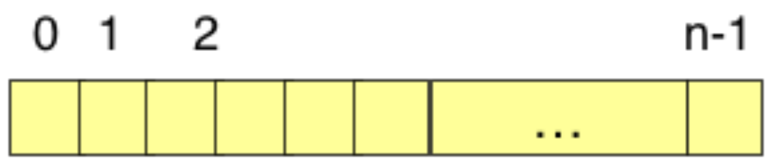


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

Free-Space Management

- Bit vector (*n* blocks)



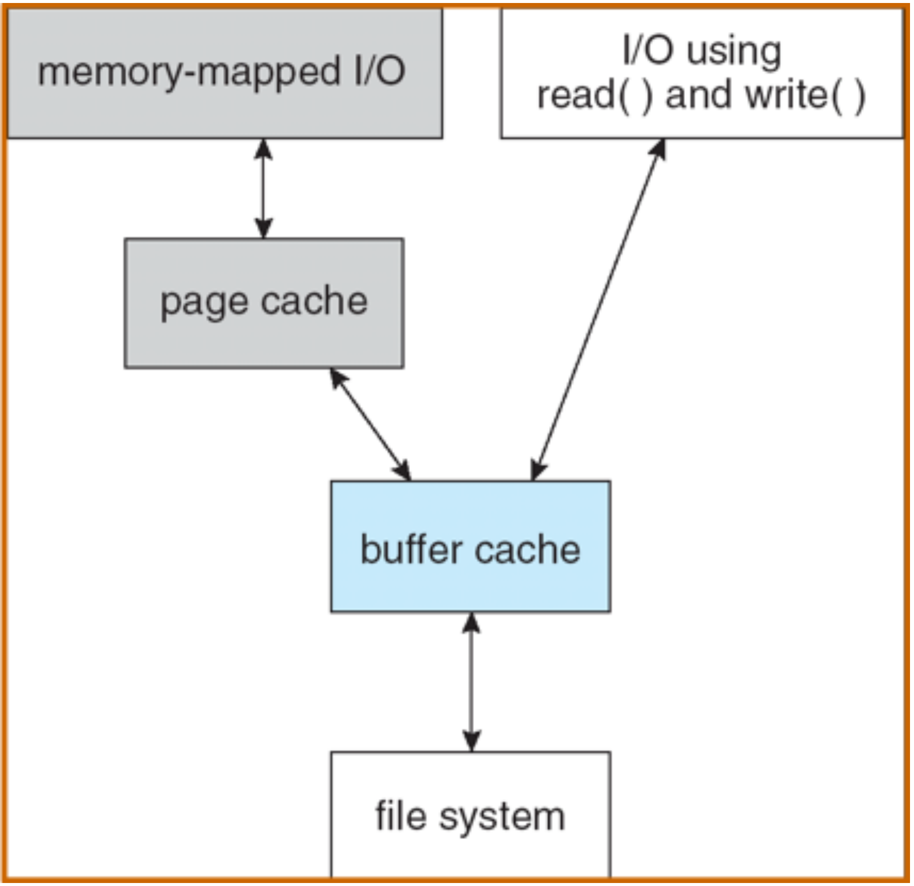
$$\text{bit}[i] = \begin{cases} 1 \Rightarrow \text{block}[i] \text{ free} \\ 0 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

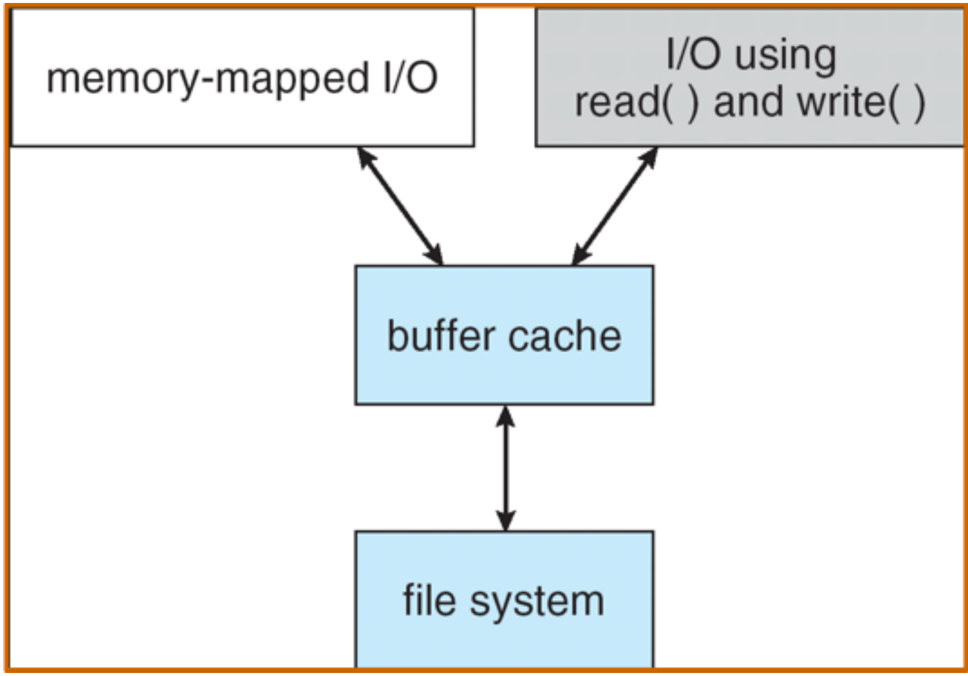
Block number calculation (finding the first free block)

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Bit map 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)

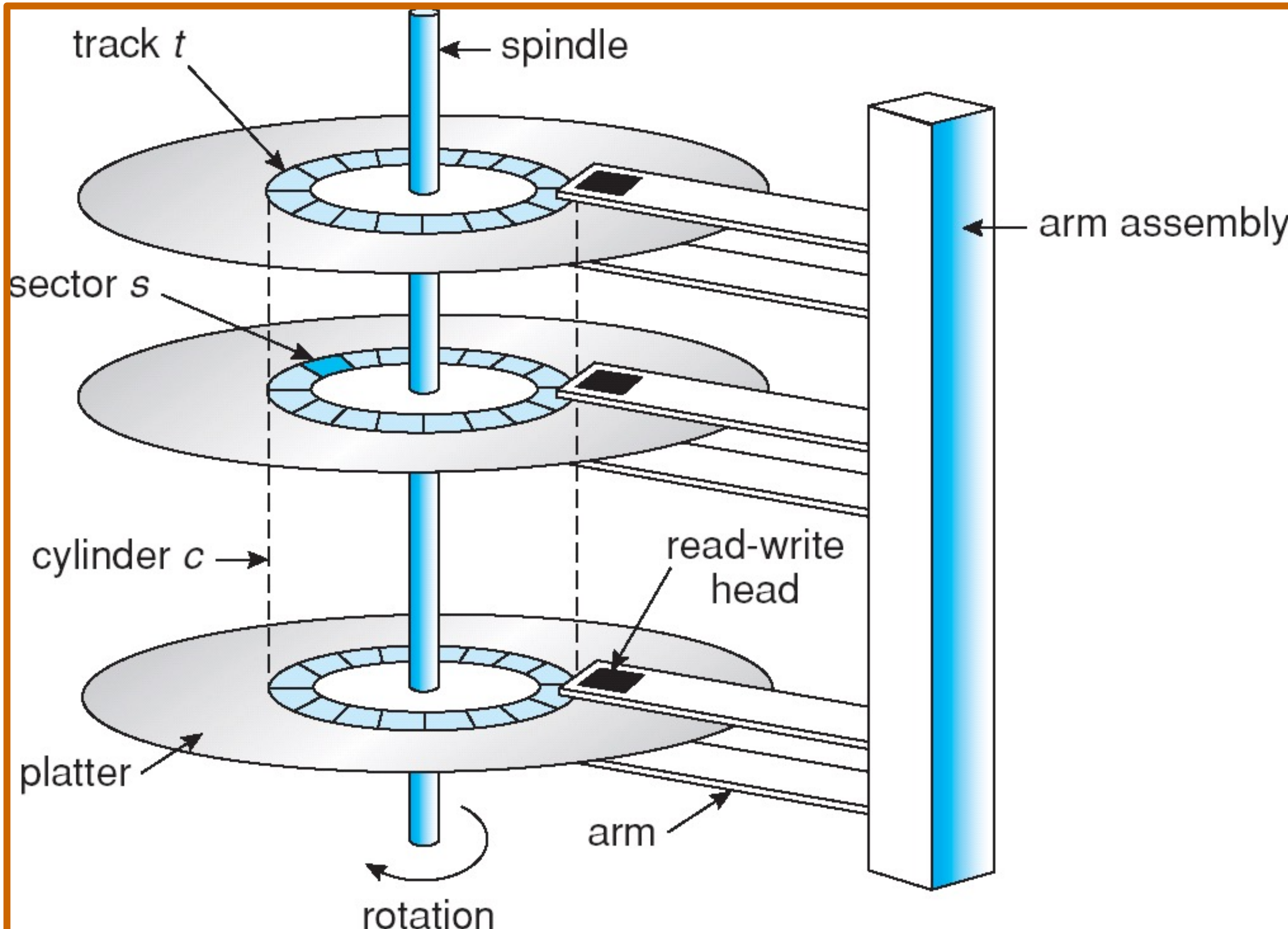
Block/Page Buffer



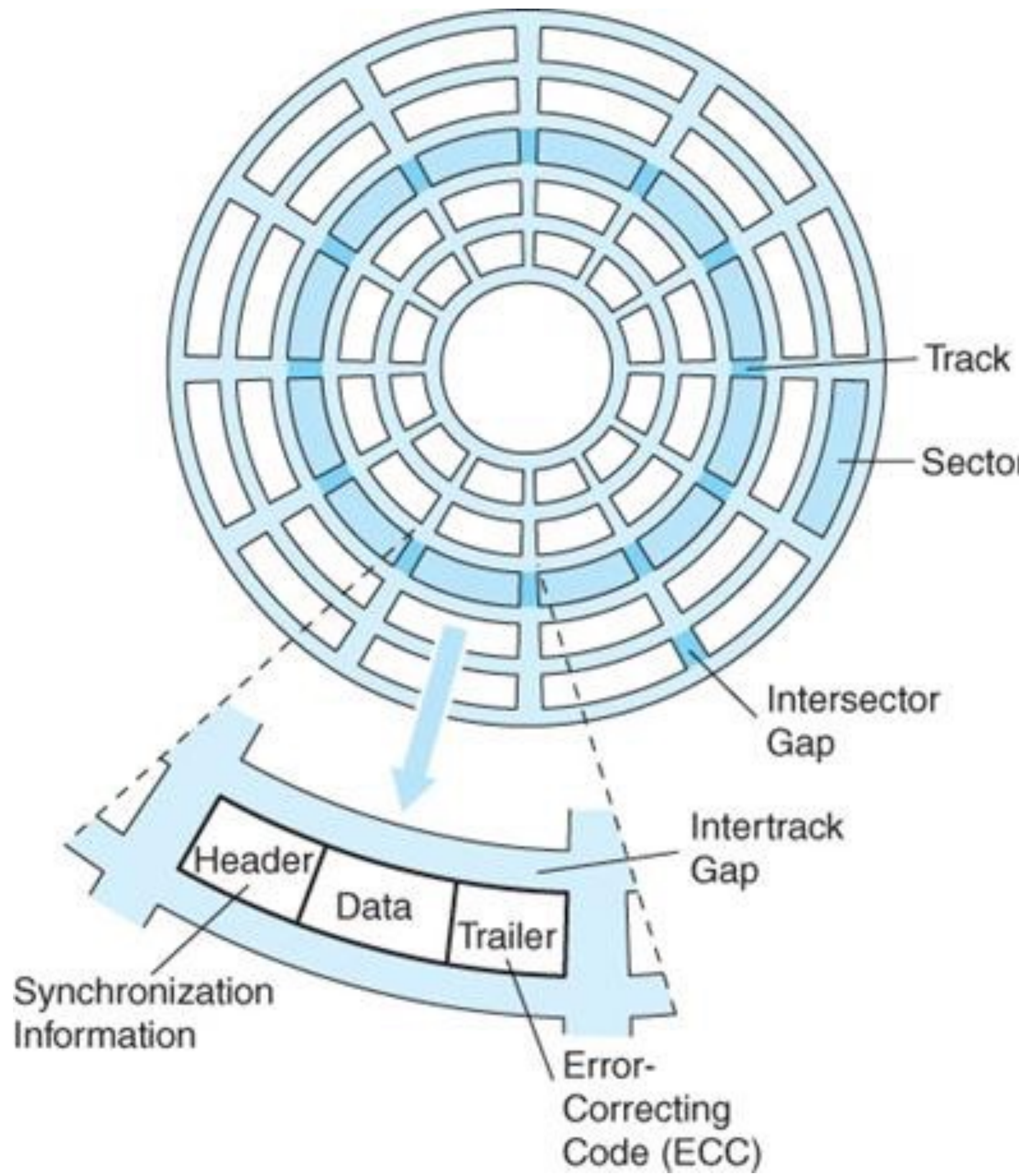


Chapter 12: Mass Storage System

- Disk

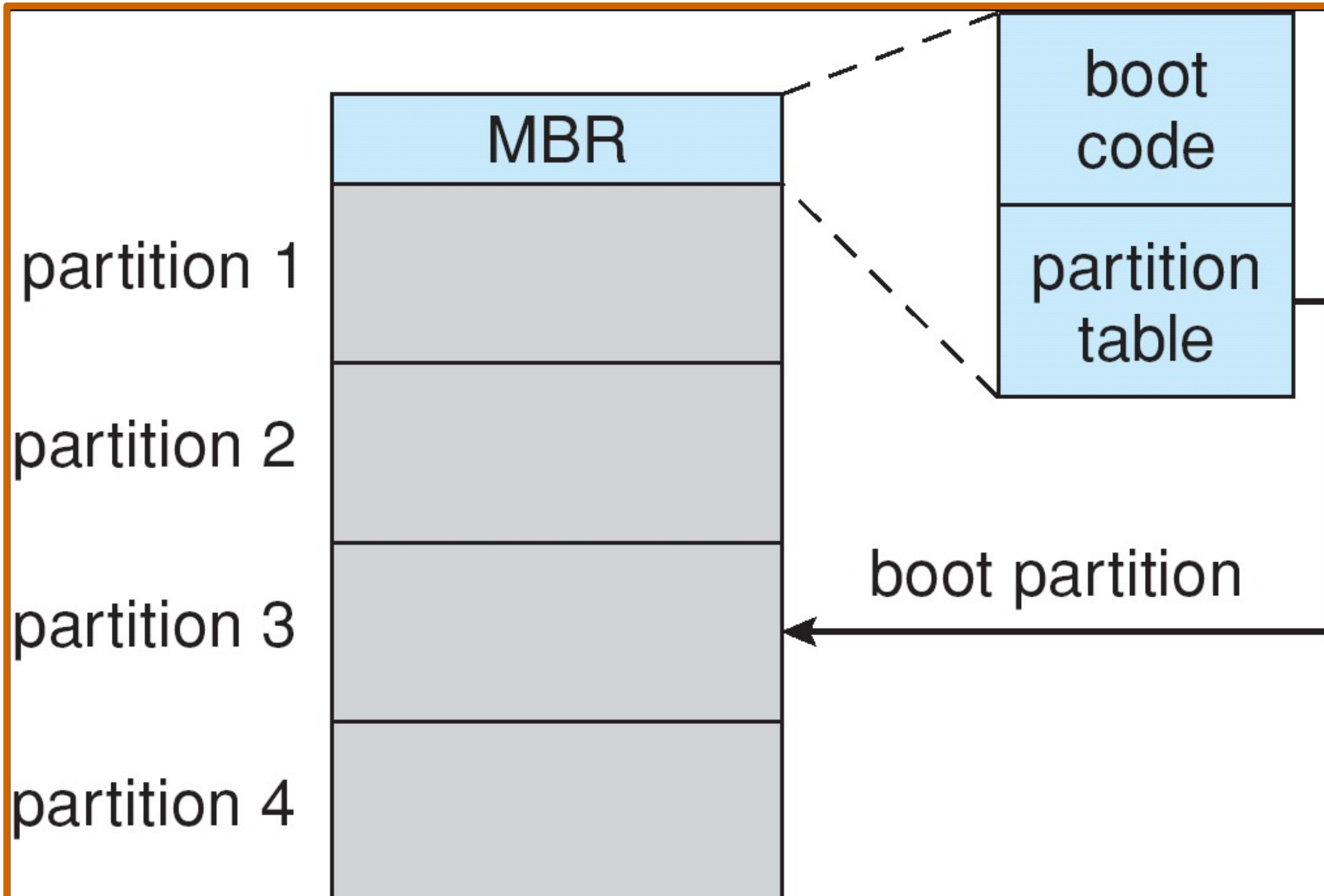


Disk Sector



Track
Sector
Intersector
Gap
Intertrack
Gap
Header
Data
Trailer
Synchronization
Information
Error-
Correcting
Code (ECC)

MBR


MBR
boot
code
partition
table
partition 1
partition 2
partition 3
partition 4
boot partition

Access Cost

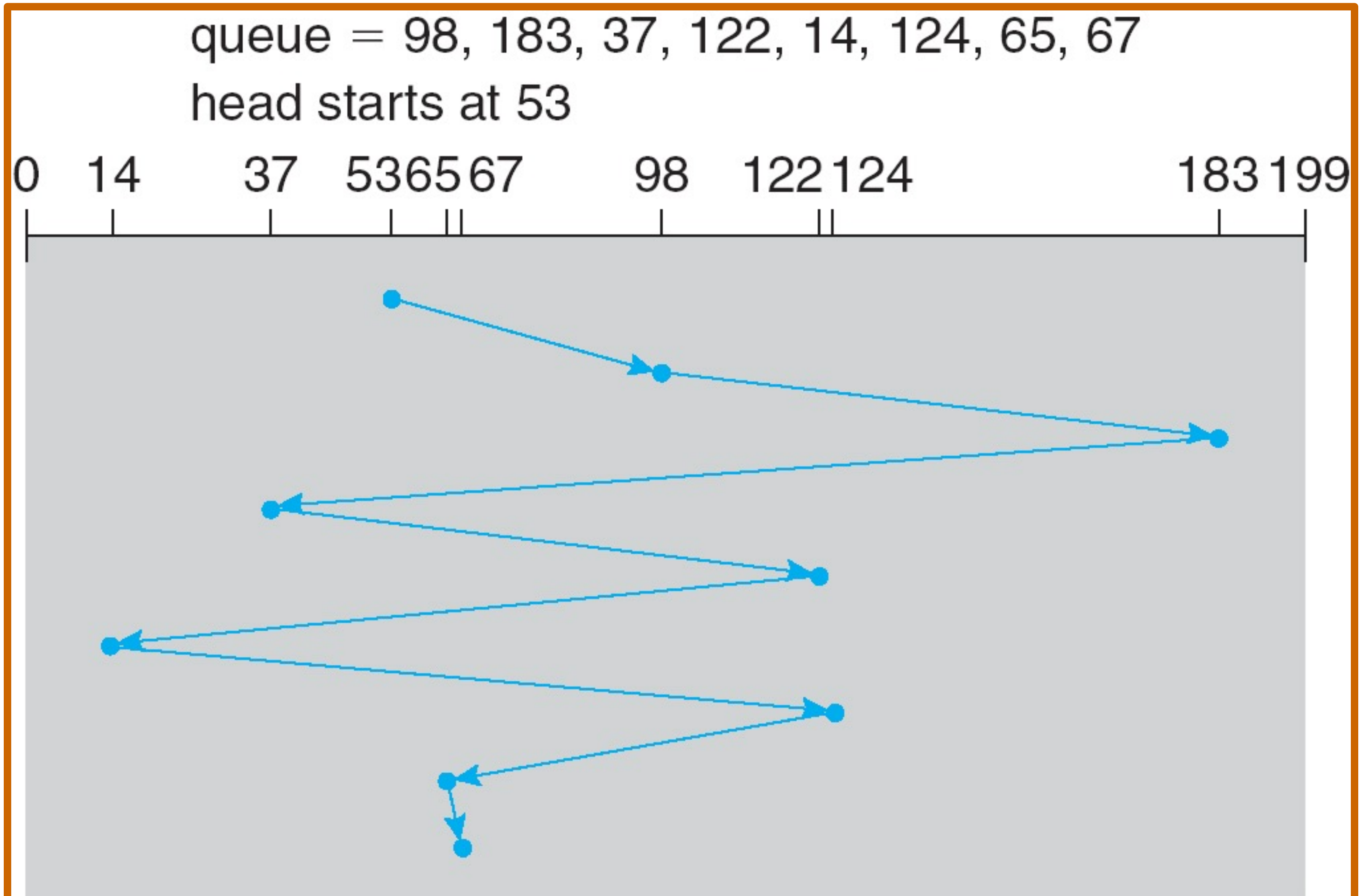
- Time to access (read/write) a disk block:
  - *seek time* (moving arms to position disk head on track)
  - *rotational delay* (waiting for block to rotate under head)
  - *transfer time* (actually moving data to/from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - As of 2010, a typical 7200 RPM desktop HDD has a "disk-to-buffer" data transfer rate up to 1030 Mbit/s
  - Key to lower I/O cost: reduce seek/rotation delays! Hardware vs. software solutions?

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - *Seek time* is the time for the disk are 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.
- Minimize seek time
- Metric: Seek time ≈ seek distance
- 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.

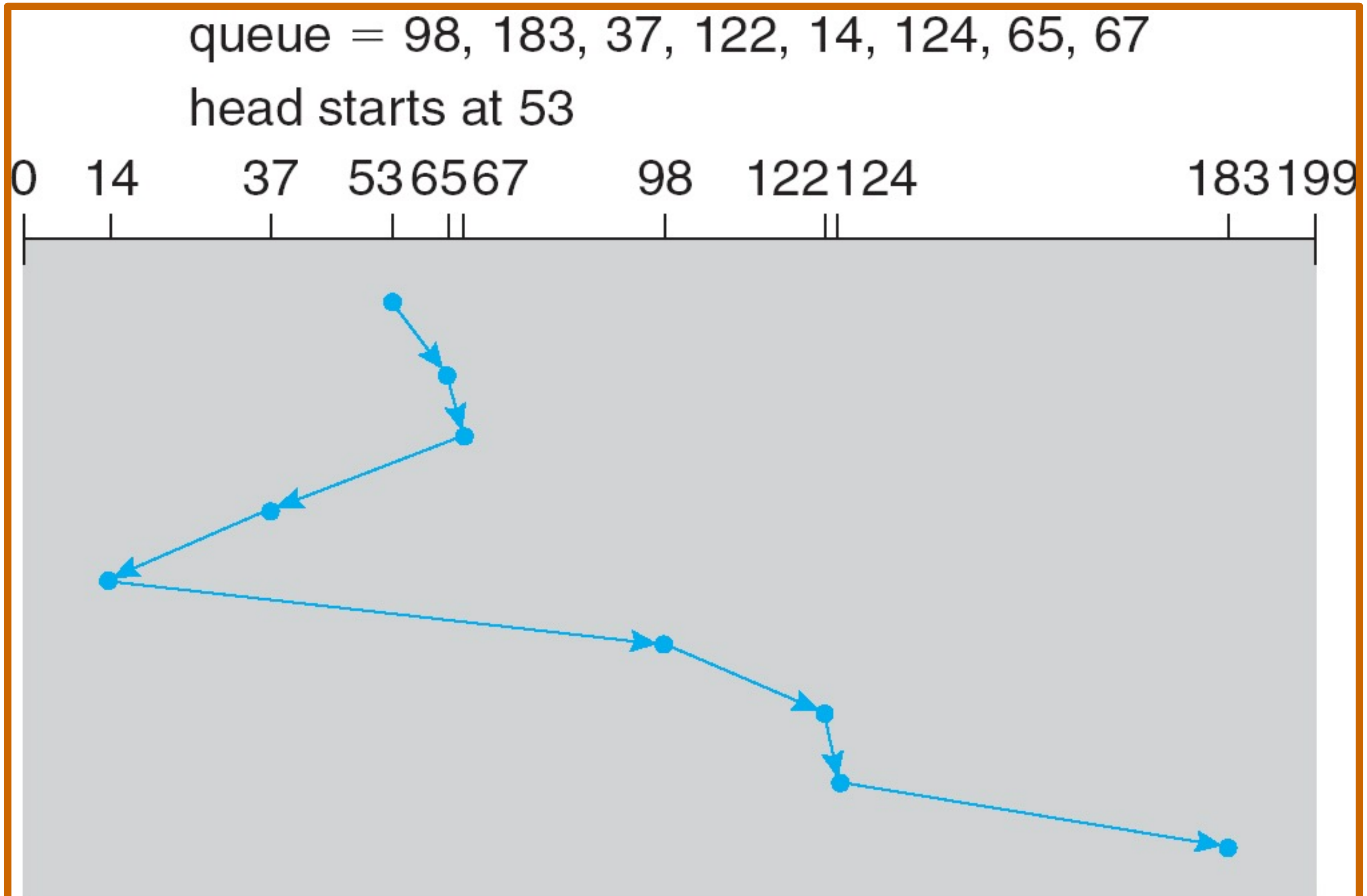
FCFS

- Total 640



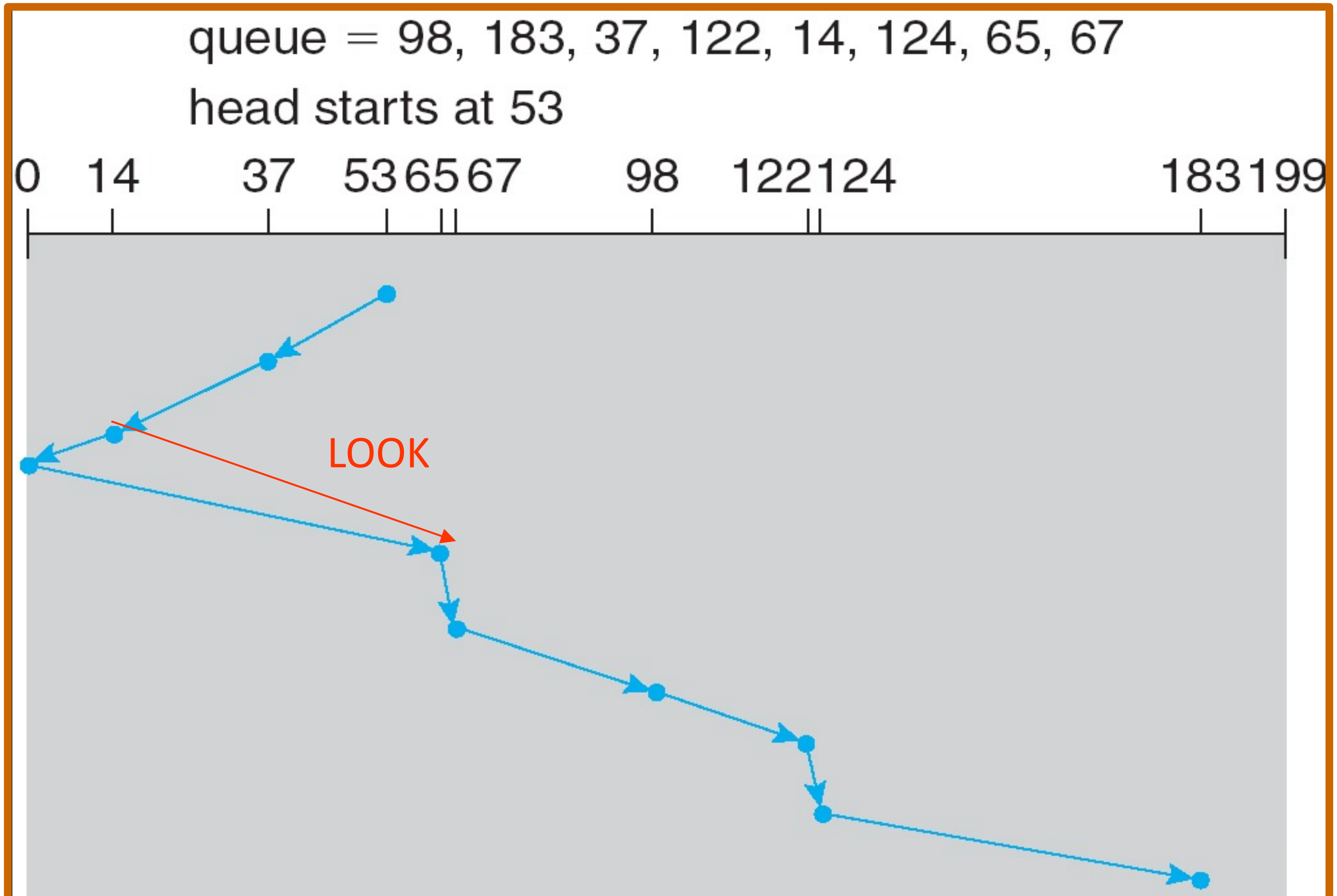
Shortest-seek-time-first (SSTF)

- Total 236

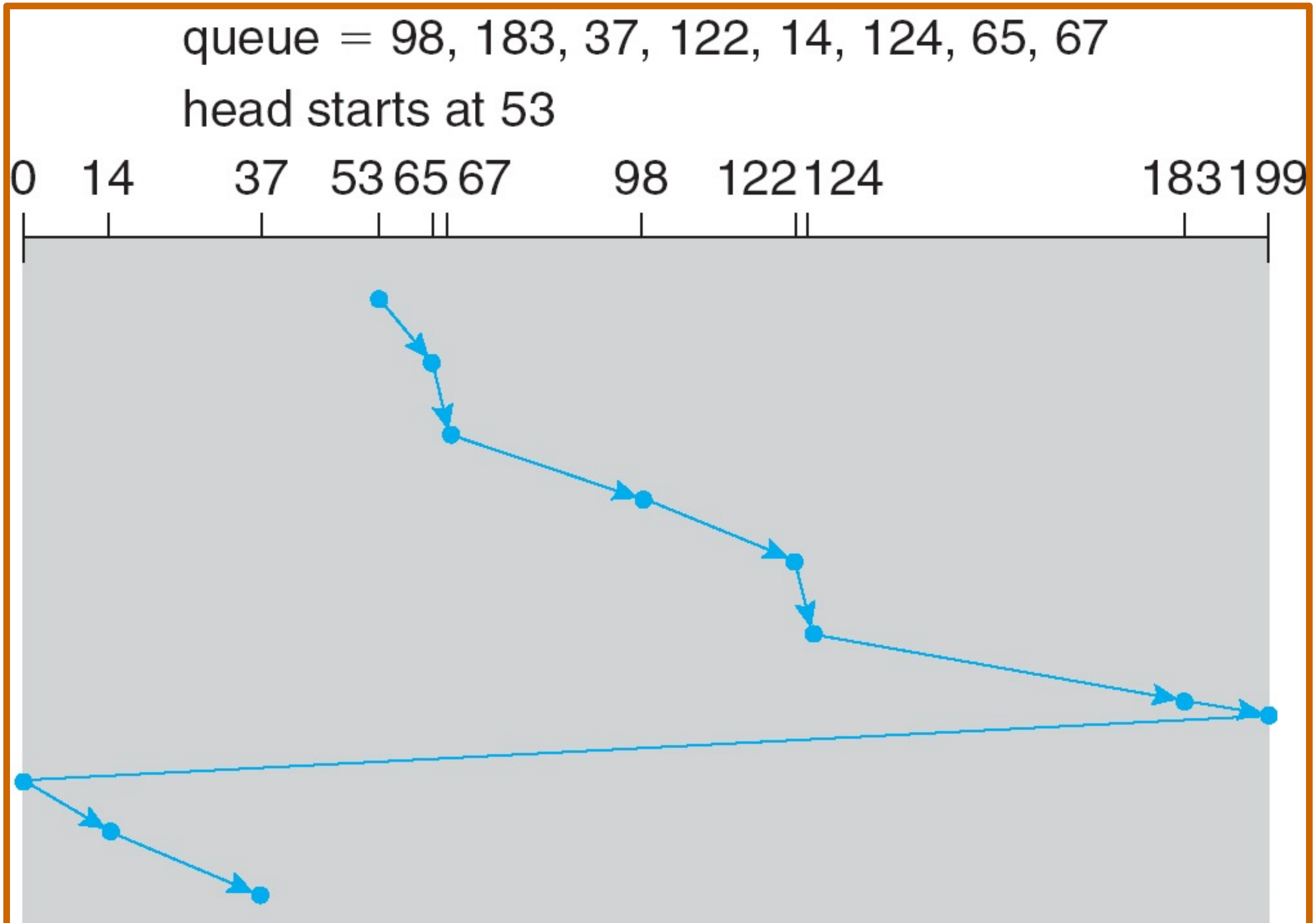


SCAN (elevator) Algorithm

- Total 208 (with look)



C-SCAN (with look)

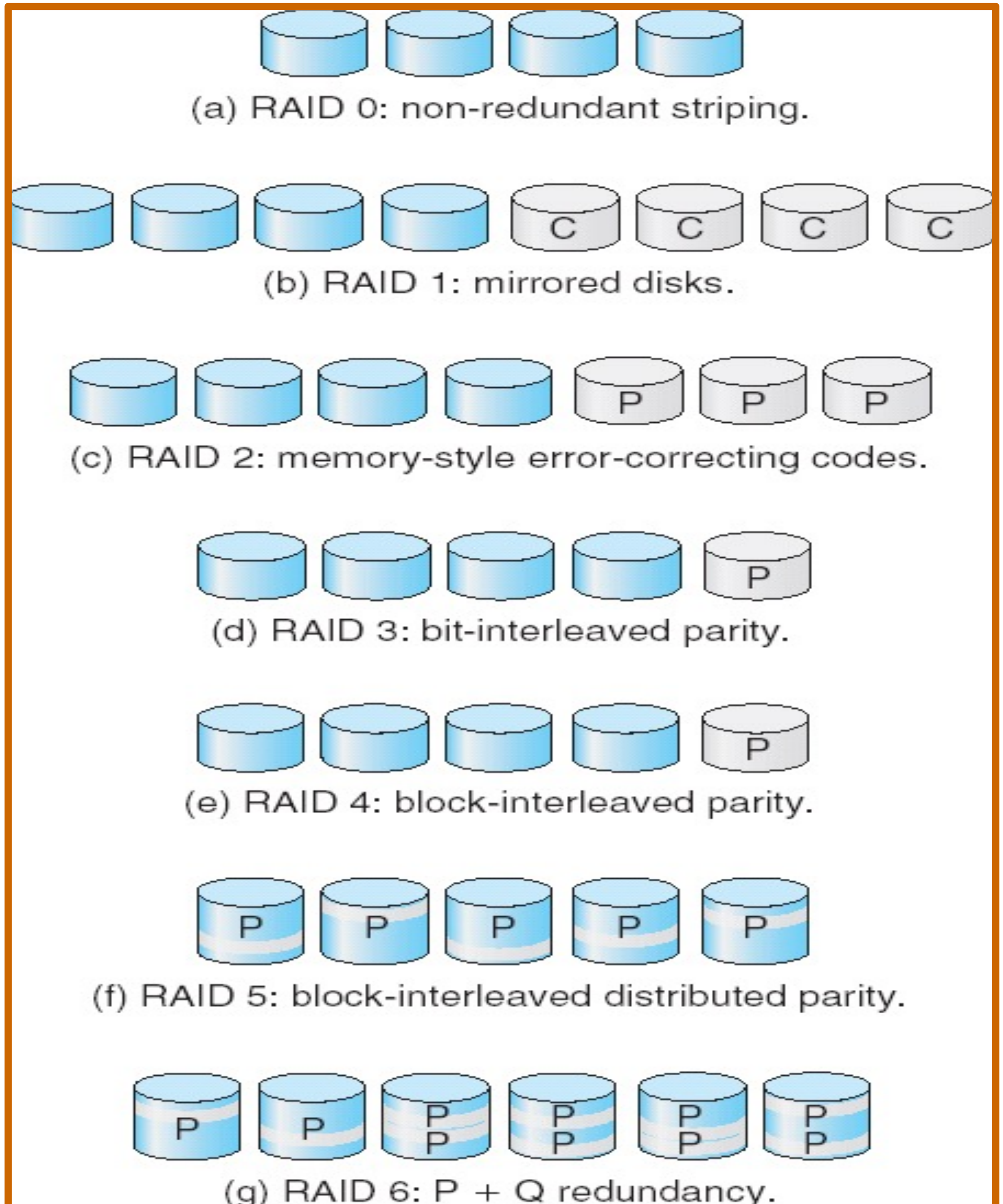


Raid (redundant arrays of inexpensive disks)

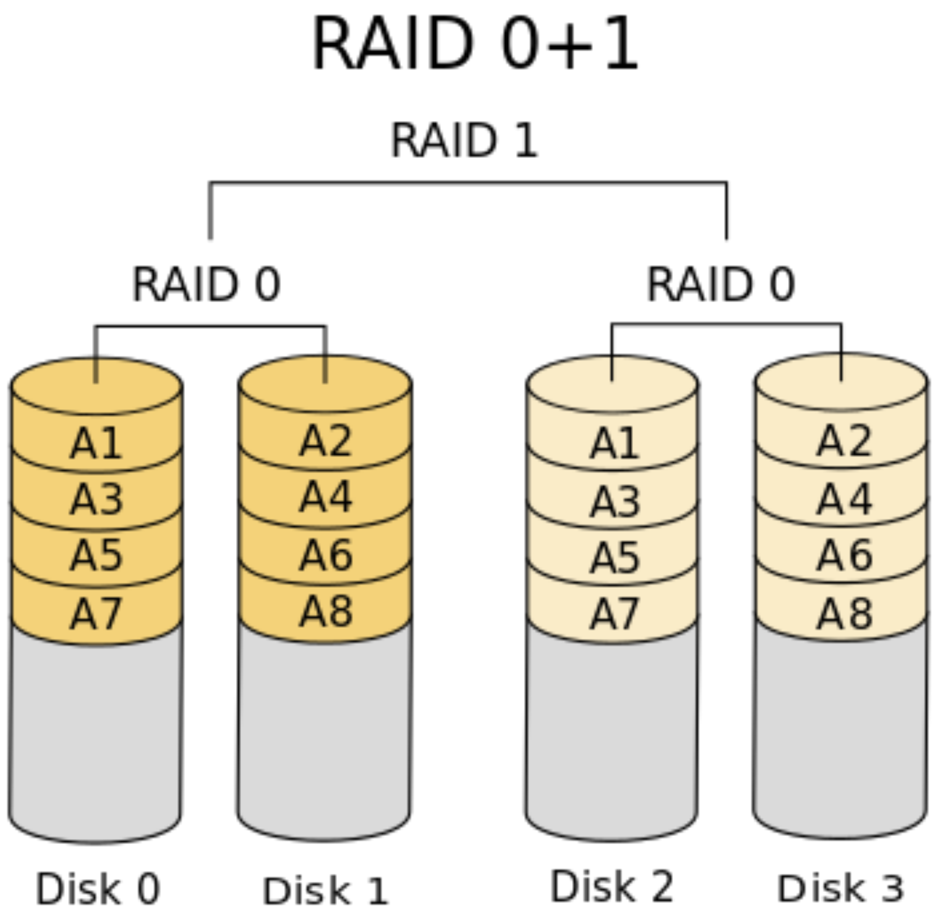
- Disk striping uses a group of disks as one storage unit.
  - Bit-level Striping
  - Block-level Striping – different blocks of a file are striped

Disk 1	Disk 2	Disk 3	Disk 4
File 1, byte 1	File 1, byte 2	File 1, byte 3	File 1, byte 4
File 1, byte 5	File 1, byte 6	File 1, byte 7	File 2, byte 1
File 2, byte 2	File 3, byte 1	File 3, byte 2	File 4 byte 1
File 4, byte 2	File 4, byte 3	*And so on…*	

Raid 0-6

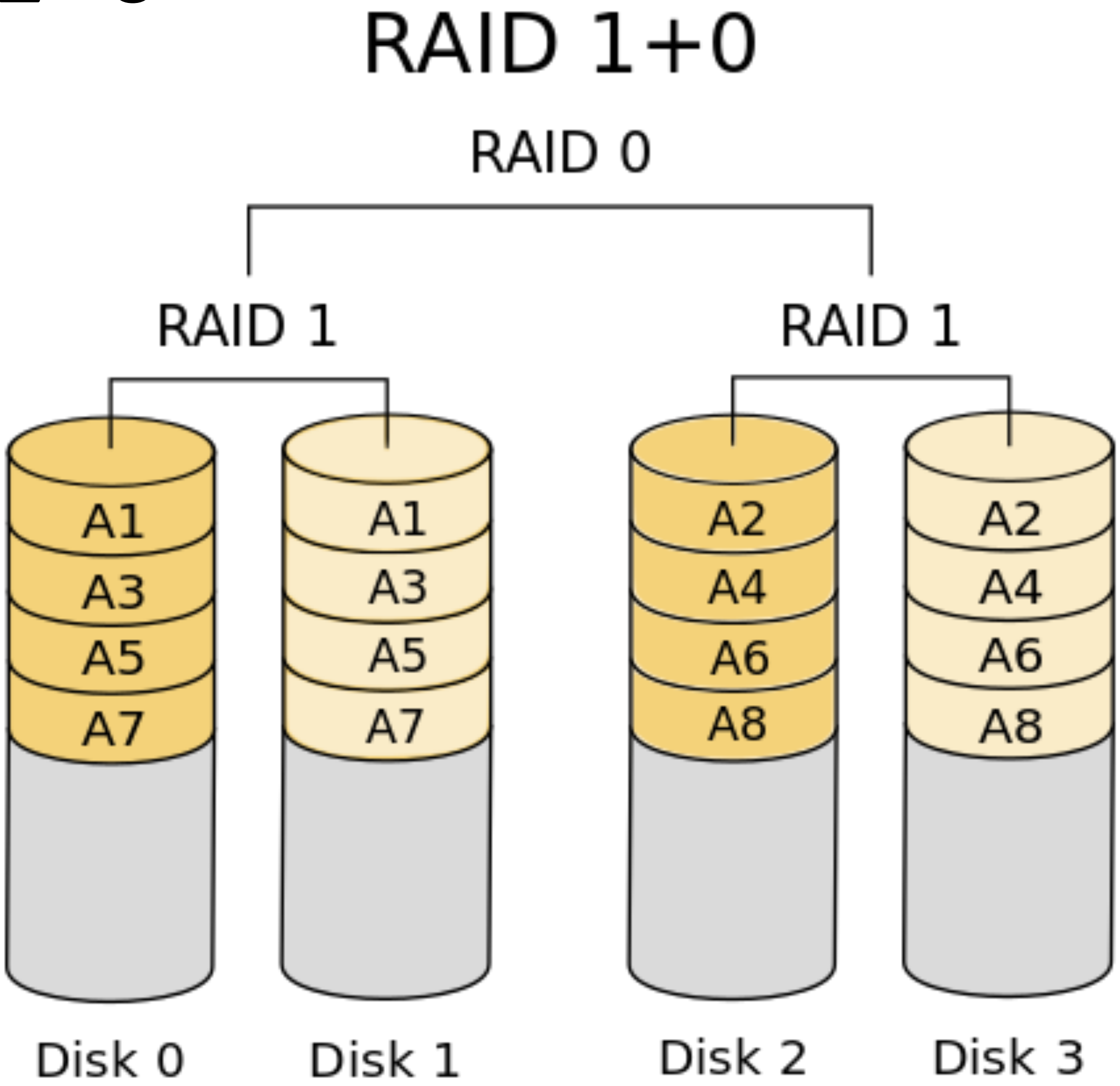

(a) RAID 0: non-redundant striping.
C
C
C
C
(b) RAID 1: mirrored disks.
P
P
P
(c) RAID 2: memory-style error-correcting codes.
P
(d) RAID 3: bit-interleaved parity.
P
(e) RAID 4: block-interleaved parity.
P
P
P
P
P
(f) RAID 5: block-interleaved distributed parity.
P
P
P
P
P
P
P
P
P
P
(g) RAID 6: P + Q redundancy.

Raid 0+1

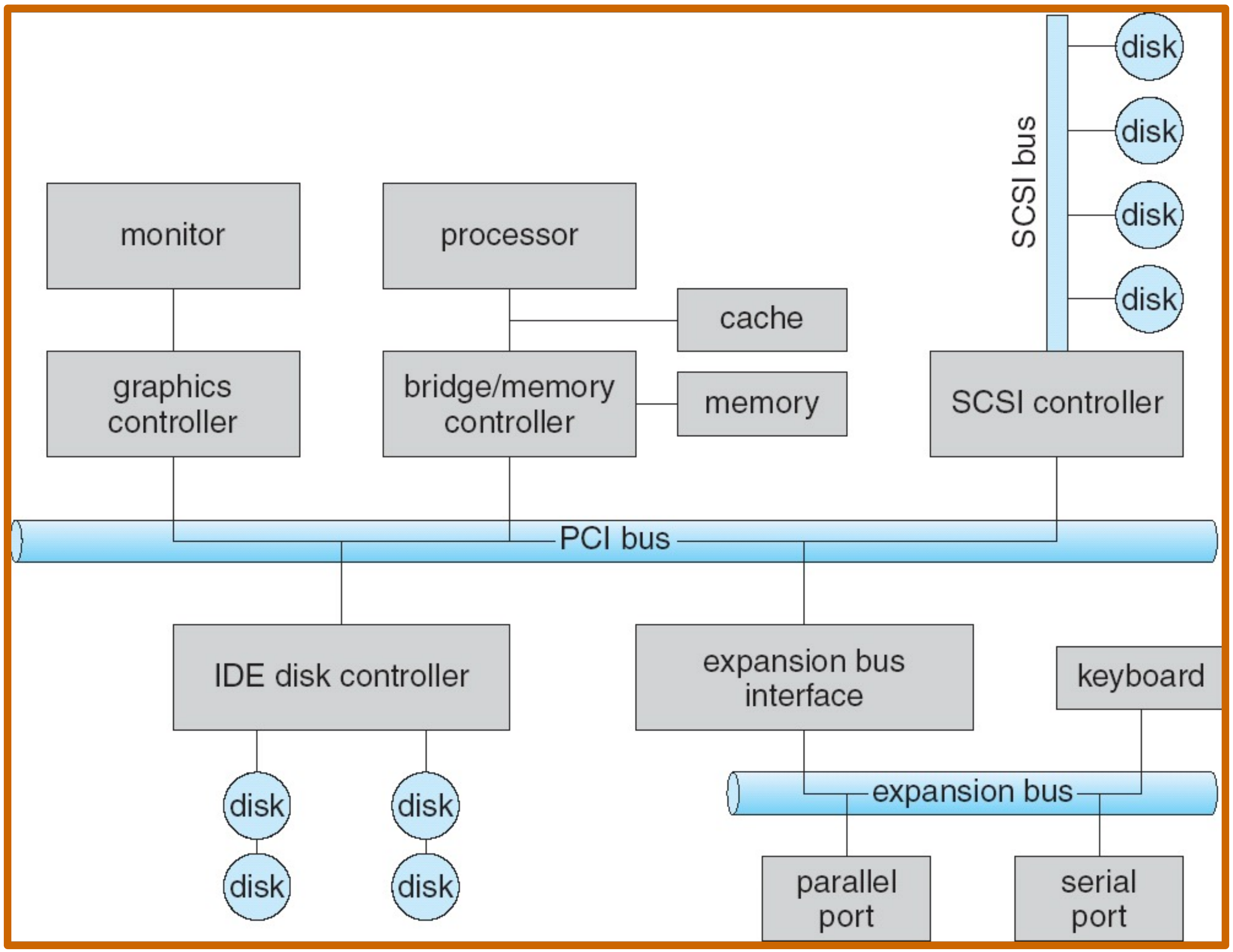


RAID 0+1
RAID 1
RAID 0
RAID 0
A1
A3
A5
A7
A2
A4
A6
A8
A1
A3
A5
A7
A2
A4
A6
A8
Disk 0
Disk 1
Disk 2
Disk 3

Raid 1+0


RAID 1+0
RAID 0
RAID 1
RAID 1
A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8
Disk 0
Disk 1
Disk 2
Disk 3

Chapter 13: I/O Systems



Device Ports

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

I/O Port Register

- **Data-in**: read by the host to get input
- **Data-out**: written by the host to send output
- **Status**: device status read by the host
- **Control**: written by the host to start a command or change the mode of a device

Old Style : Polling

- (Refer to textbook p.499)Determines state of device
  - command-ready
  - busy
  - Error
- **Busy-wait** cycle to wait for I/O from device

Repeatedly reading the **status** register until the busy bit becomes clear.
Inefficient!!

New Style: Interruption

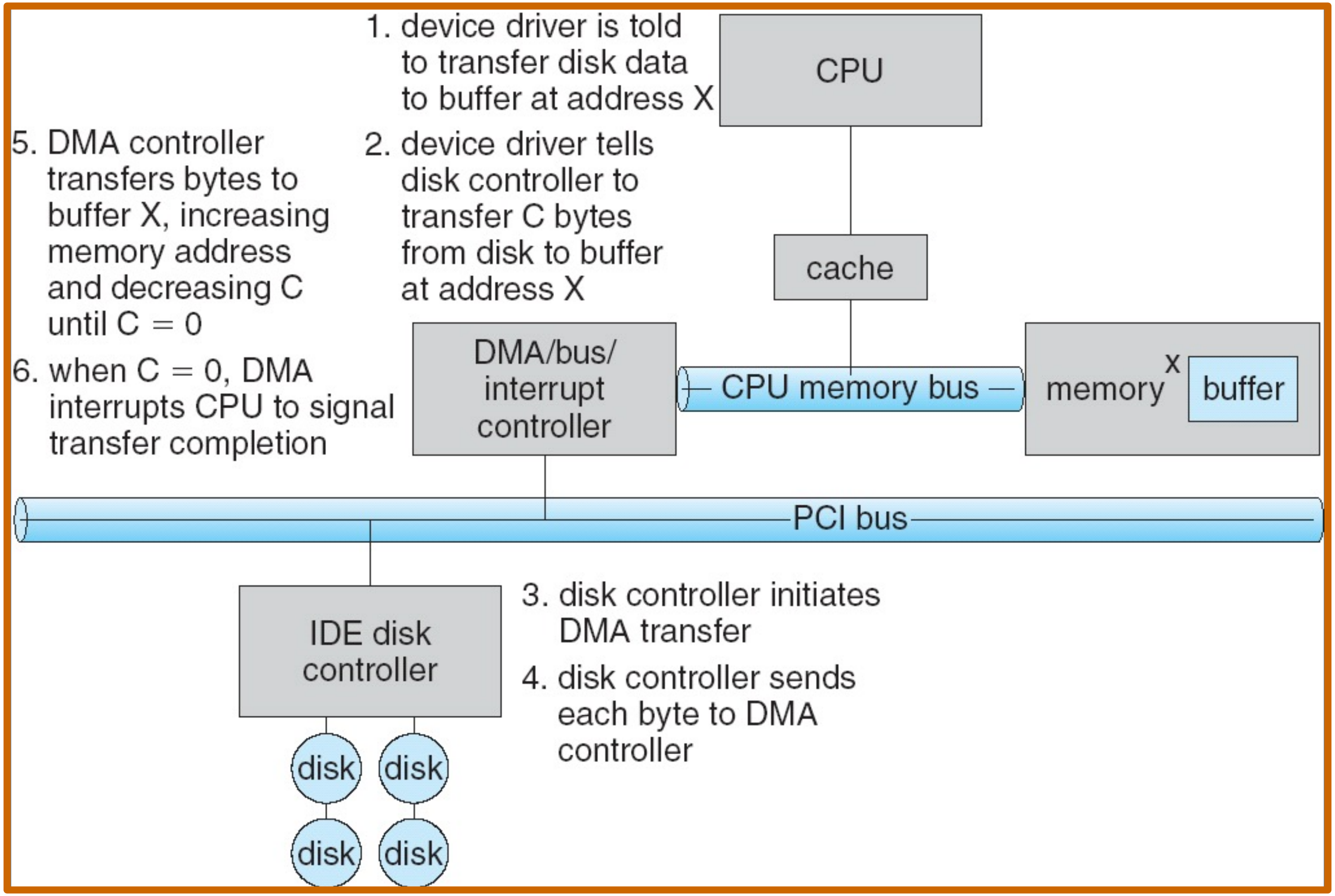
- CPU **Interrupt-request line** triggered by I/O device
- **Interrupt handler** receives interrupts
- **Maskable** to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some **nonmaskable**
- Interrupt mechanism also used for exceptions

Signal and Interrupt

- **Signals** are standardized messages sent to a running program to trigger specific behavior, such as quitting or error handling. They are a limited form of inter-process communication (IPC)

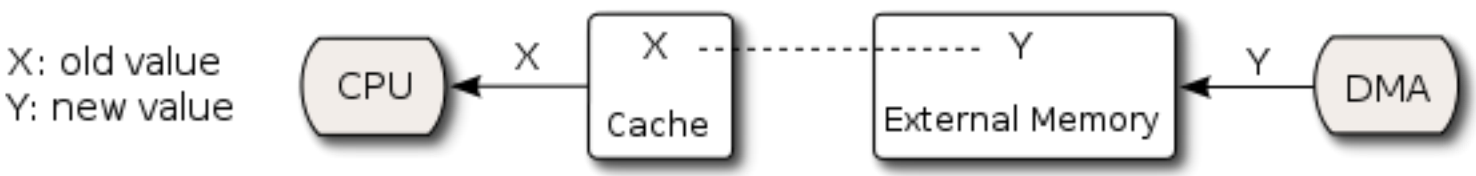
SIGINT	2	Interrupt a process (used by **Ctrl-C**)
SIGHUP	1	Hang up or shut down and restart process
SIGKILL	9	Kill the process (cannot be ignored or caught elsewhere)
SIGTERM	15	Terminate signal, (can be ignored or caught)
SIGTSTP	20	Stop the terminal (used by **Ctrl-z**)
SIGSTOP	19	Stop execution (cannot be caught or ignored)

Direct Memory Access



DMA Cache Problem

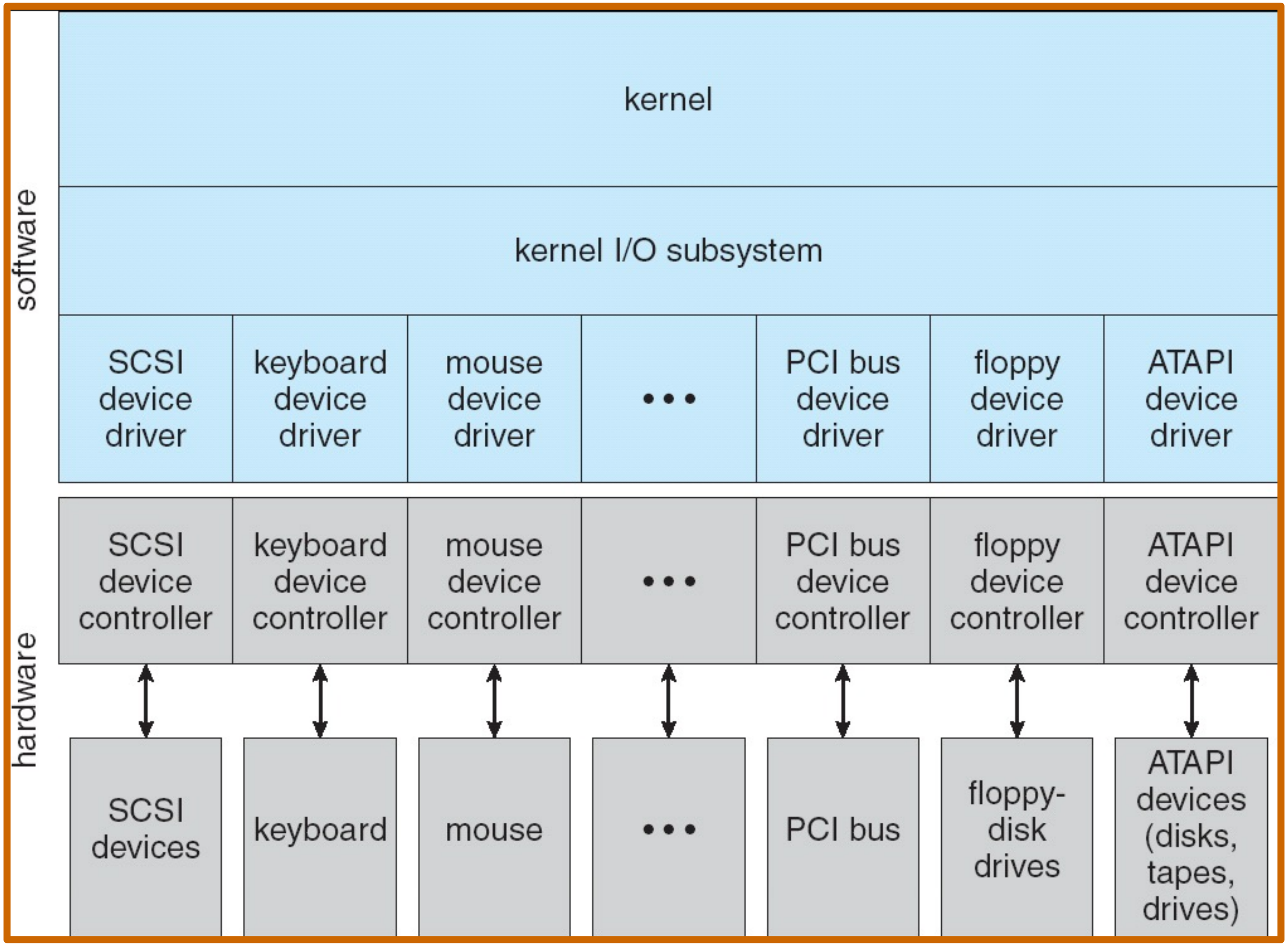
- DMA updates a value in memory which has been modified by CPU in cache



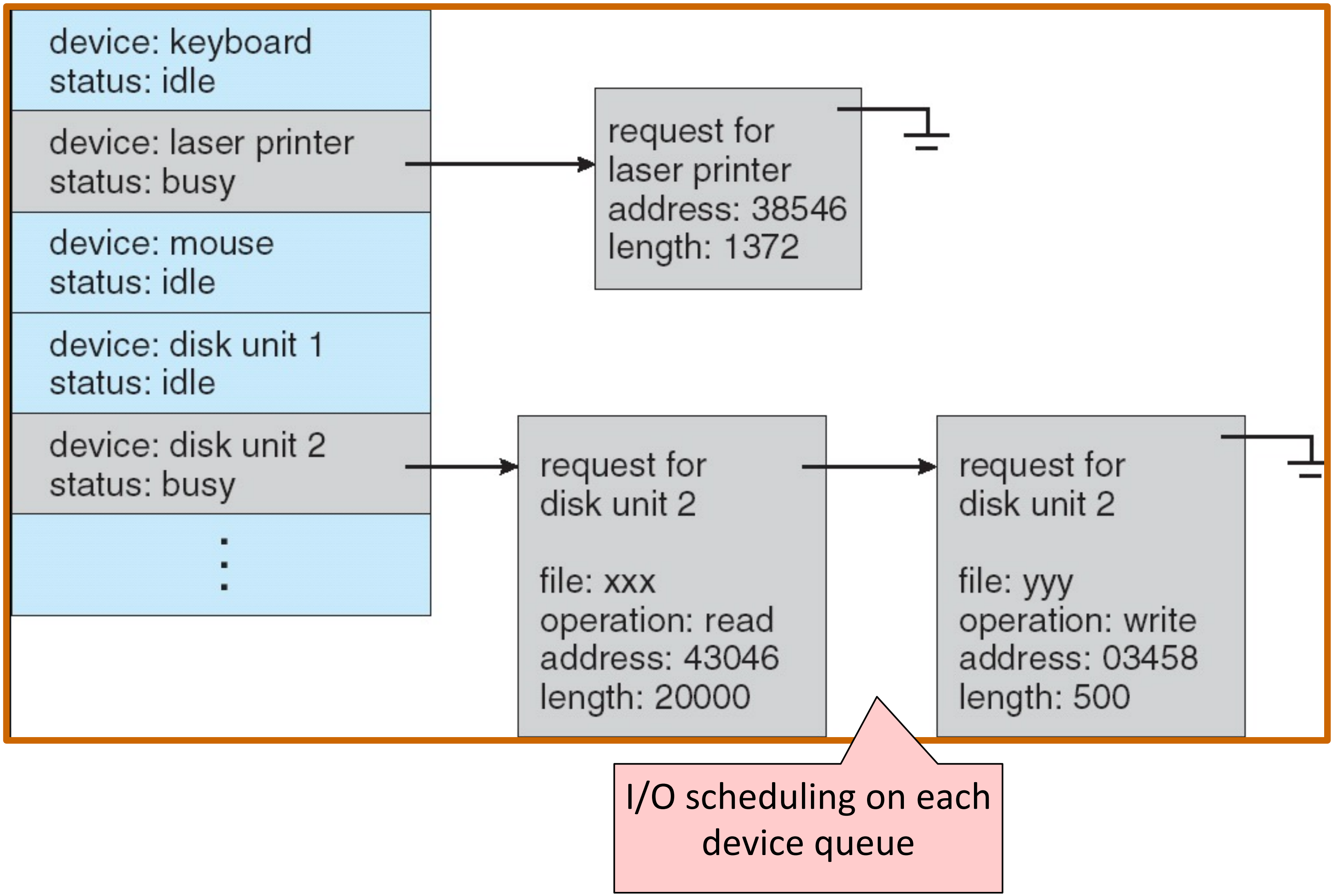
Types of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Device Drivers



Device Status Table



I/O Mode

	Blocking	Non-blocking
Synchronous	Read/write	Read/wirte (O_NONBLOCK)
Asynchronous	I/O multiplexing (select/poll)	AIO

Examination Info

- 40 Selections
- 5 fill-in
- 1 mem + 1 scheduling + 1 synchronization