ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

OPERATING SYSTEMS & PARALLEL COMPUTING

Disk Management

ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

FileSystem Interface

Contents

- File Concept
- Access Methods
- Directory Structures
- File System Mounting
- File Sharing
- Protection

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

File Concept

- Contiguous logical address space.
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program
- Contents defined by file's creator:
  - Many types:
    - Consider text file, source file, executable file

File Structure

- None – sequence of words, bytes.
- Simple record structure:
  - Lines
  - Fixed length
  - Variable length
- Complex Structures:
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters.
- Who decides:
  - Operating system
  - Program

Basic file attributes

- **Name** – only information kept in human-readable form.
- **Identifier** – unique tag (number) identifies file within system.
- **Type** – needed for systems that support different types.
- **Location** – pointer to file location on device.
- **Size** – current file size.
- **Protection** – controls who can do reading, writing, executing.
- **Time, date, and user identification** – data for protection, security, and usage monitoring.
- Information about files are kept in the directory structure, which is maintained on the disk.
- Many variations, including extended file attributes such as file checksum.
- Information kept in the directory structure.

More file attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

File Type Extensions (1)

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

File Type Extensions (2)

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes com-pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Common File Operations

- Create
- Read/Write – at read/write pointer location.
- Seek: Reposition within file.
- Delete
- Append/Truncate
- Set/Get Attributes
- Open(F_i): search the directory structure on disk for entry F_i, and move the content of entry to memory.
- Close (F_i): move the content of entry F_i in memory to directory structure on disk.

Open Files

- Several pieces of data needed to manage open files:
  - Open-file table: tracks open files.
  - File pointer: pointer to last read/write location, per process that has the file open.
  - File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it.
  - Disk location of the file: cache of data access information.
  - Access rights: per-process access mode information.

Access Methods

- **Sequential Access**

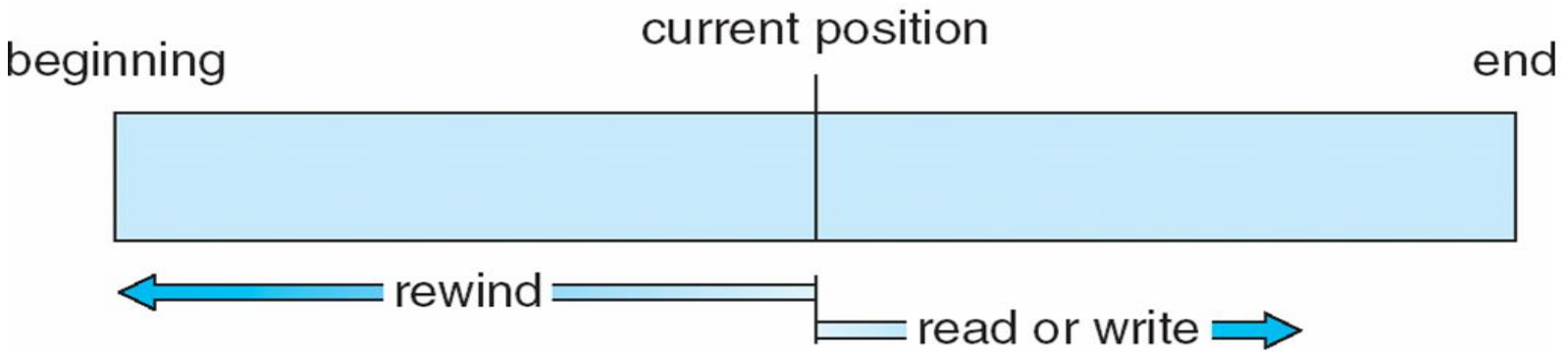
  read next
  write next
  reset
  no *read after last write*
  (rewrite)

- **Direct Access –** file is fixed length logical records

  read n
  write n
  position to n
  read next
  write next
  rewrite n

  n = relative block number

Sequential/Direct Access

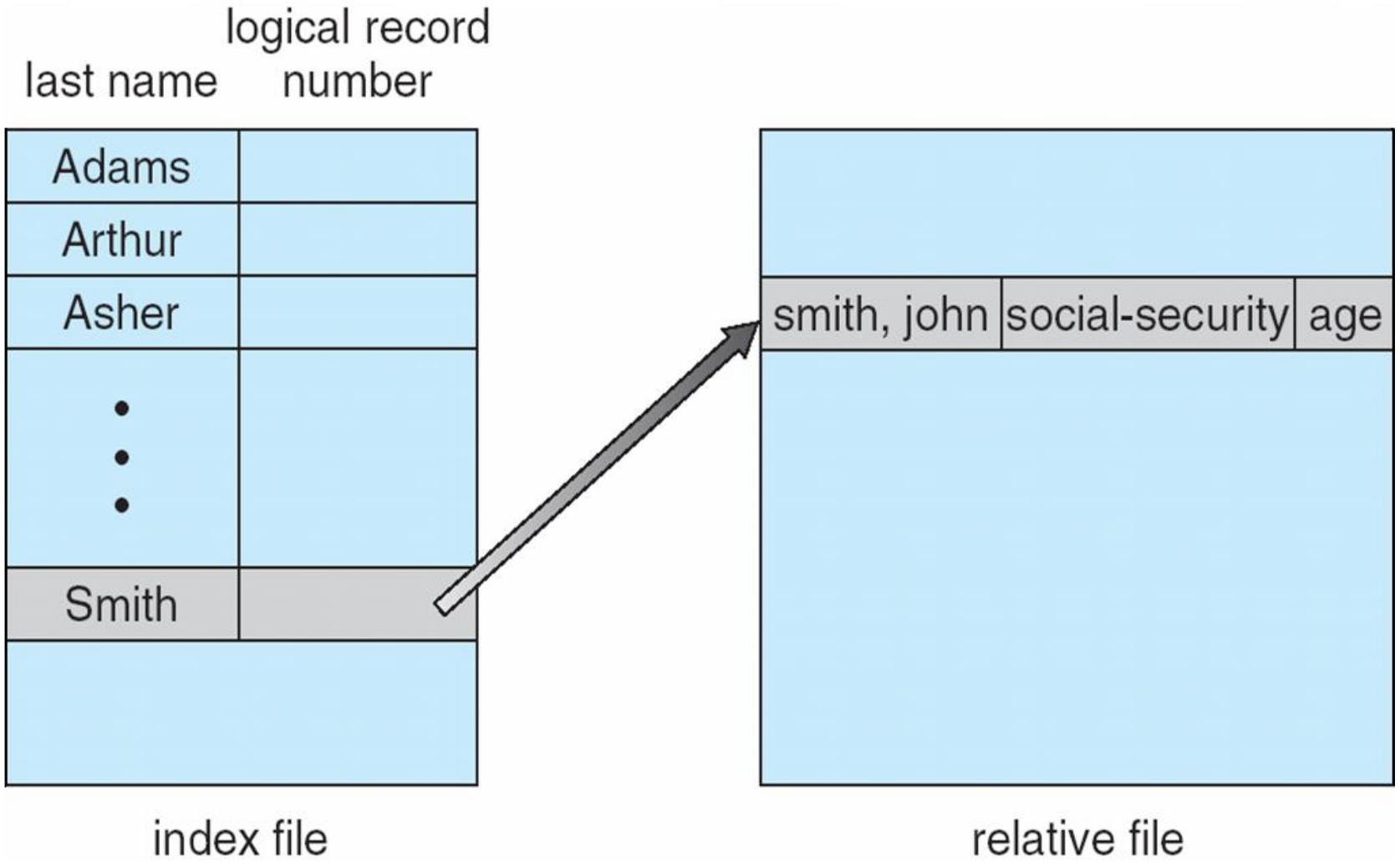


sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Other Access Methods

- Can be built on top of base methods.
- General involve creation of an index for the file.
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item).
- If too large, index (in memory) of the index (on disk).
- IBM indexed sequential-access method (ISAM):
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS.
- VMS operating system provides index and relative files as another example (see next slide).

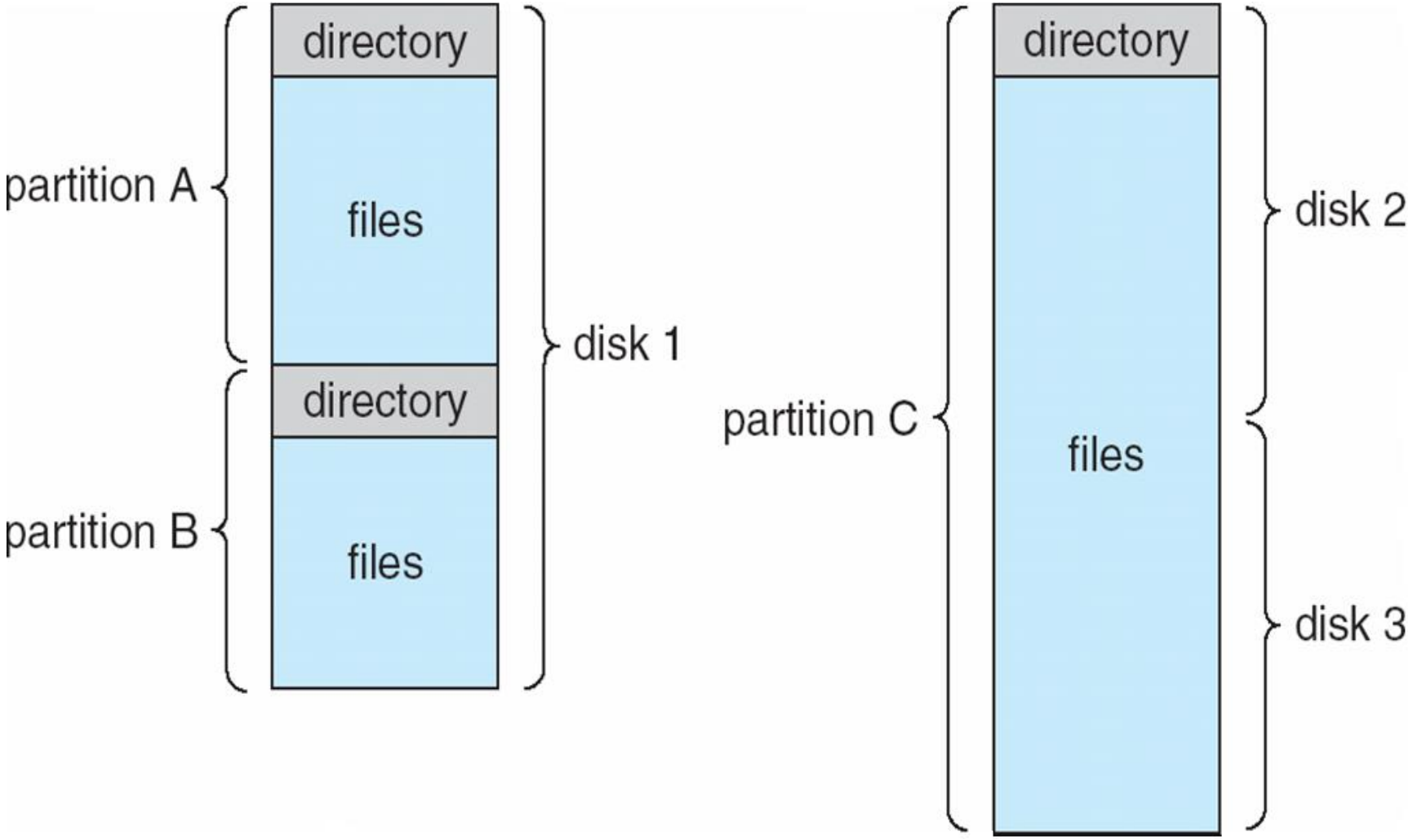
Example of Index and Relative Files



Disk Structure

- Disk can be subdivided into partitions.
- Disks or partitions can be RAID protected against failure.
- Disk or partition can be used raw – without a file system, or formatted with a file system.
- Partitions also known as minidisks, slices.
- Entity containing file system known as a volume.
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents.
- As well as general-purpose file systems, there are many special-purpose file systems, frequently all within the same operating system or computer.

A Typical File-system Organization



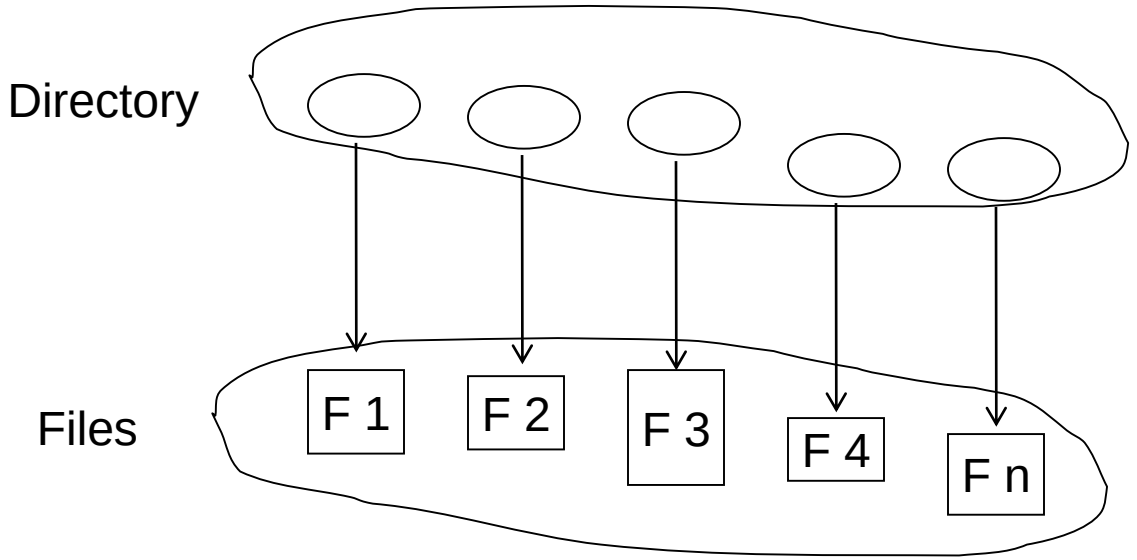
ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Types of File Systems

- We mostly talk of general-purpose file systems.
- But systems frequently have may file systems, some general- and some special- purpose.
- Consider Solaris has:
  - tmpfs – memory-based volatile FS for fast, temporary I/O
  - objfs – interface into kernel memory to get kernel symbols for debugging
  - ctfs – contract file system for managing daemons
  - lofs – loopback file system allows one FS to be accessed in place of another
  - procfs – kernel interface to process structures
  - ufs, zfs – general purpose file systems

Directory Structures

- Collection of nodes containing information about all files.



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

Information in a Directory Entry

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information

Directory Operations

- Operations performed on a directory:
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system

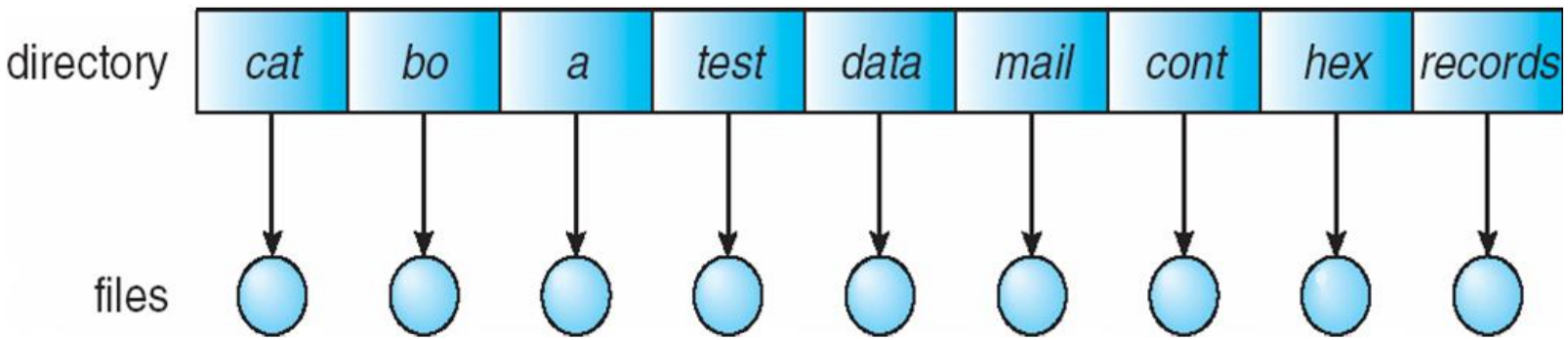
Directory Organization Characteristics

Organize the directory (logically) to obtain:

- **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, …).

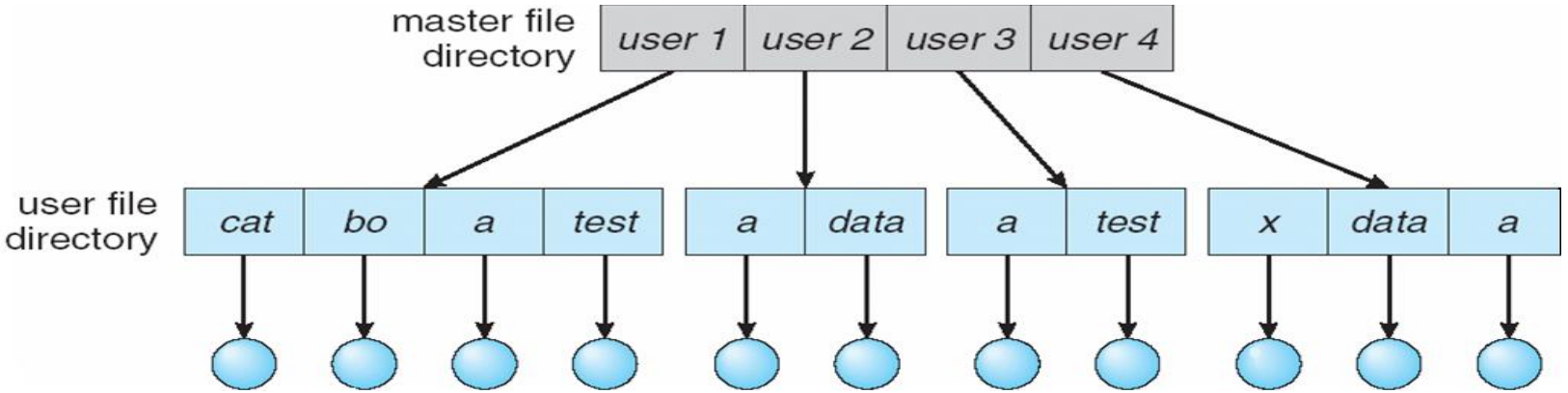
Single-Level Directory

- A single directory for all users.
- Common problems:
  - Eventual length of directory.
  - Giving unique names to files.
  - Remembering names of files.
  - Grouping of files (use file extensions).

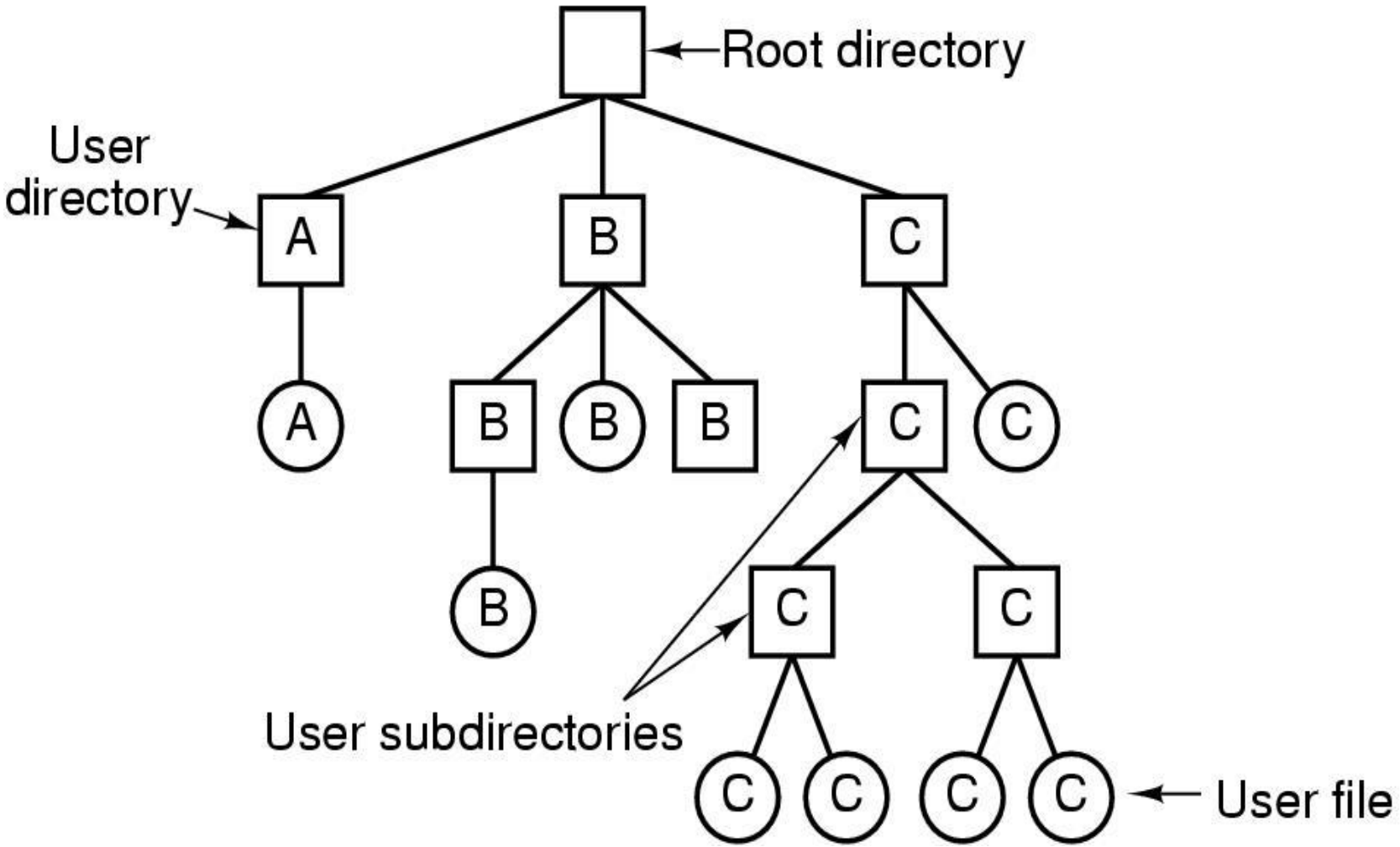


Two-Level Directory

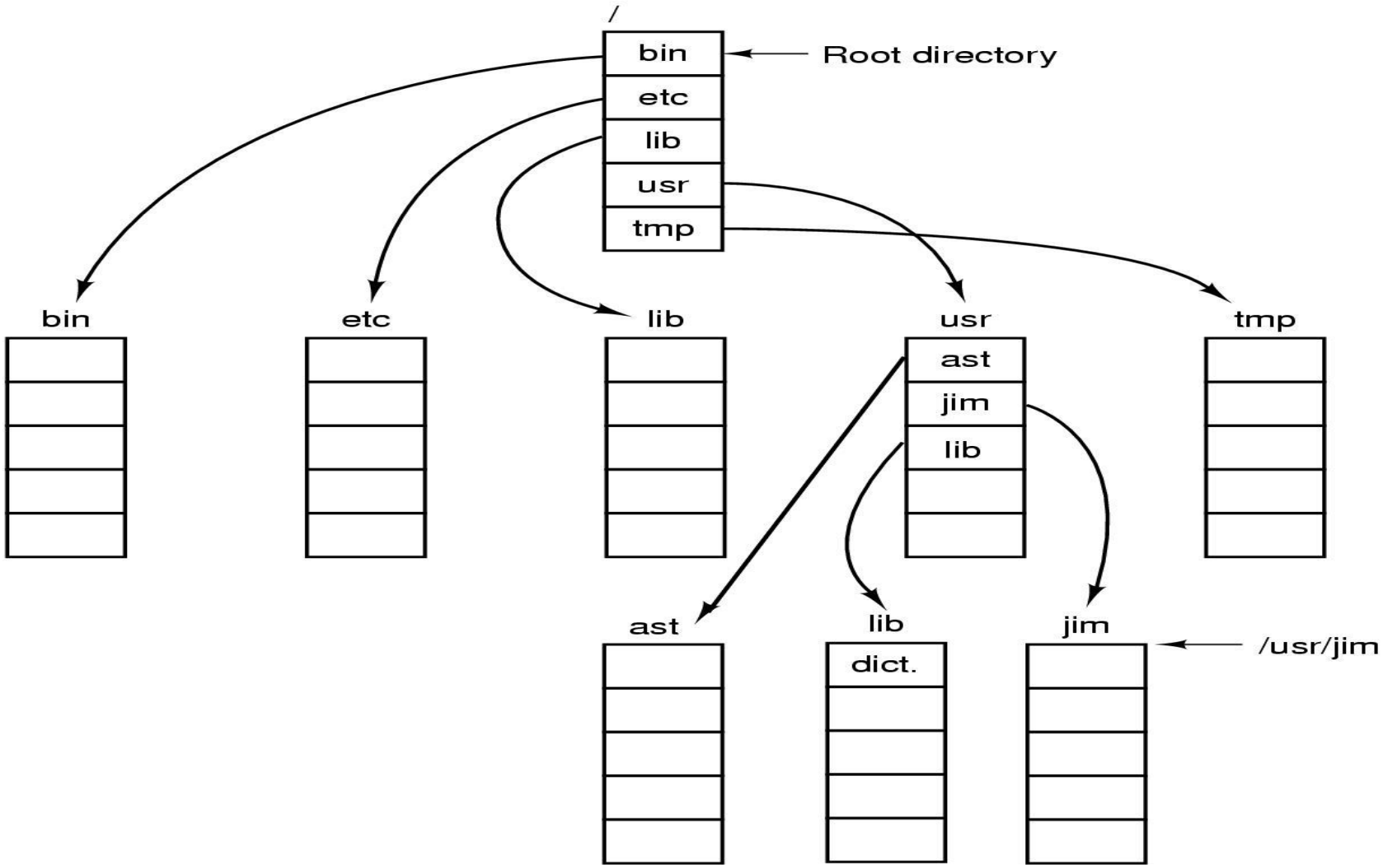
- Separate directory for each user.
- Use of path name.
- Can have the same file name for different users.
- Provides efficient searching.
- No grouping capability.
- Main problem: violates zero-one-infinity principle.



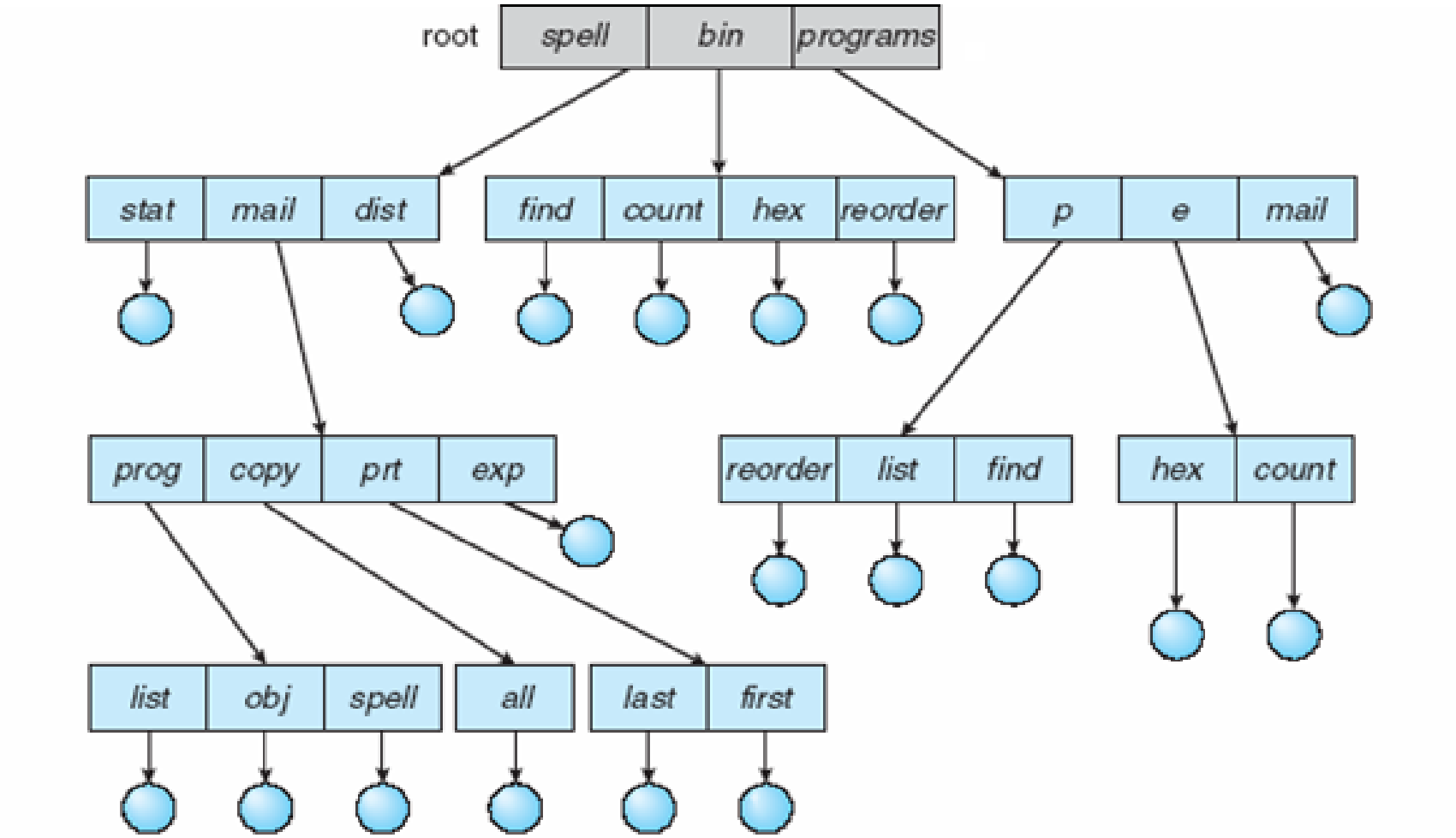
Tree-Structured Directory Components



Directory Path Names



Tree-Structured Directory



Tree-Structured Directories (1)

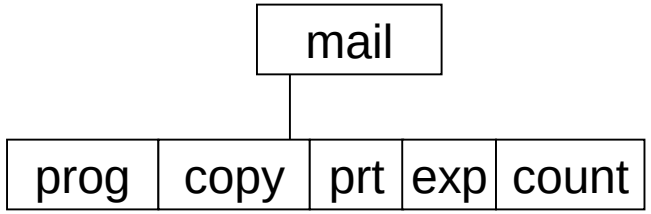
- Provides efficient searching.
- Has grouping capability.
- Current directory (working directory):
  - **cd** /spell/mail/prog
  - **type** list
- Use absolute or relative path name.

Tree-Structured Directories (2)

- Creating a new file is done in current directory.
- Delete a file: **rm** <file-name>
- Creating a new subdirectory is done in current directory: **mkdir** <dir-name>

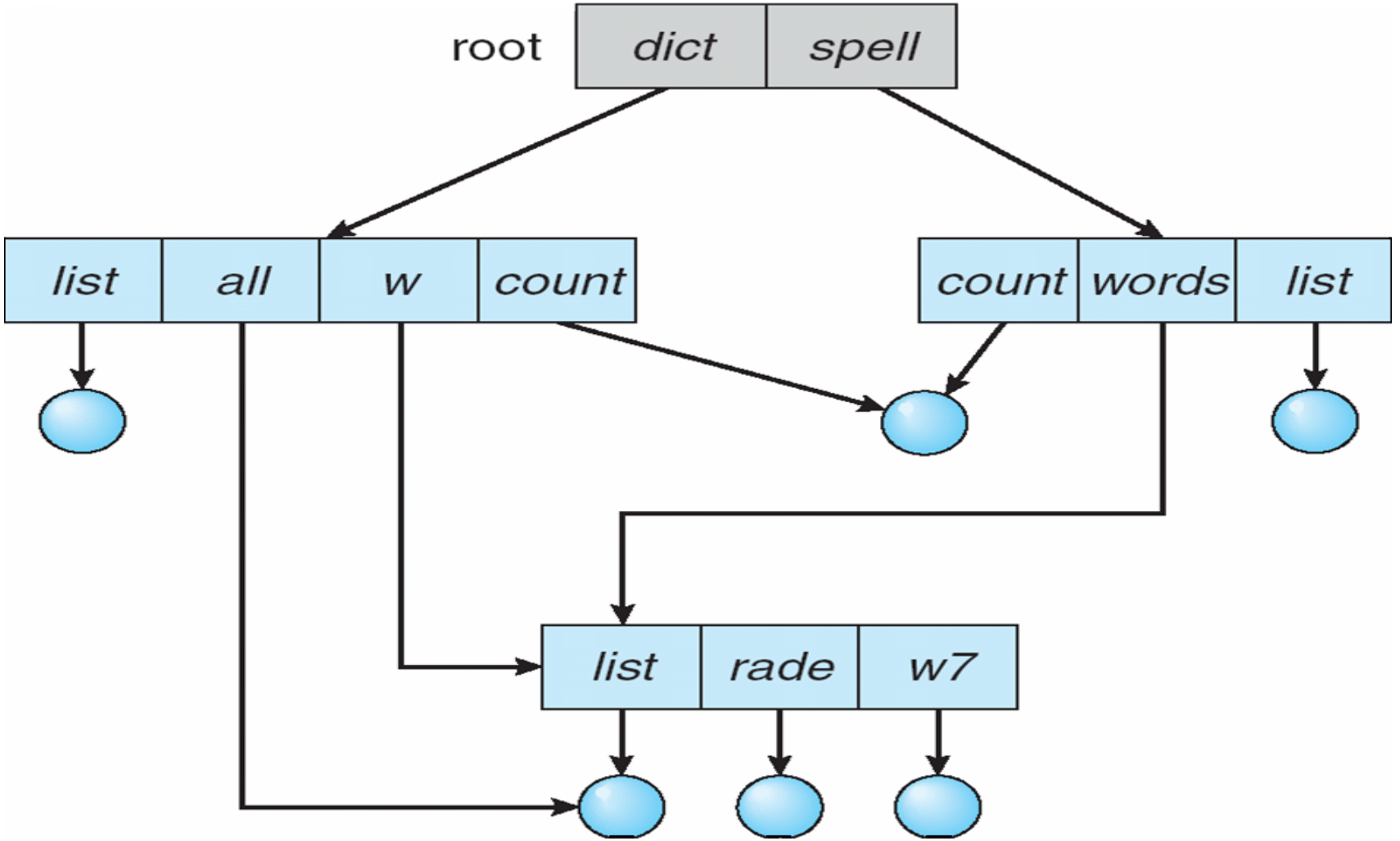
  Example: if in current directory **/mail**

  mkdir count



- Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”.

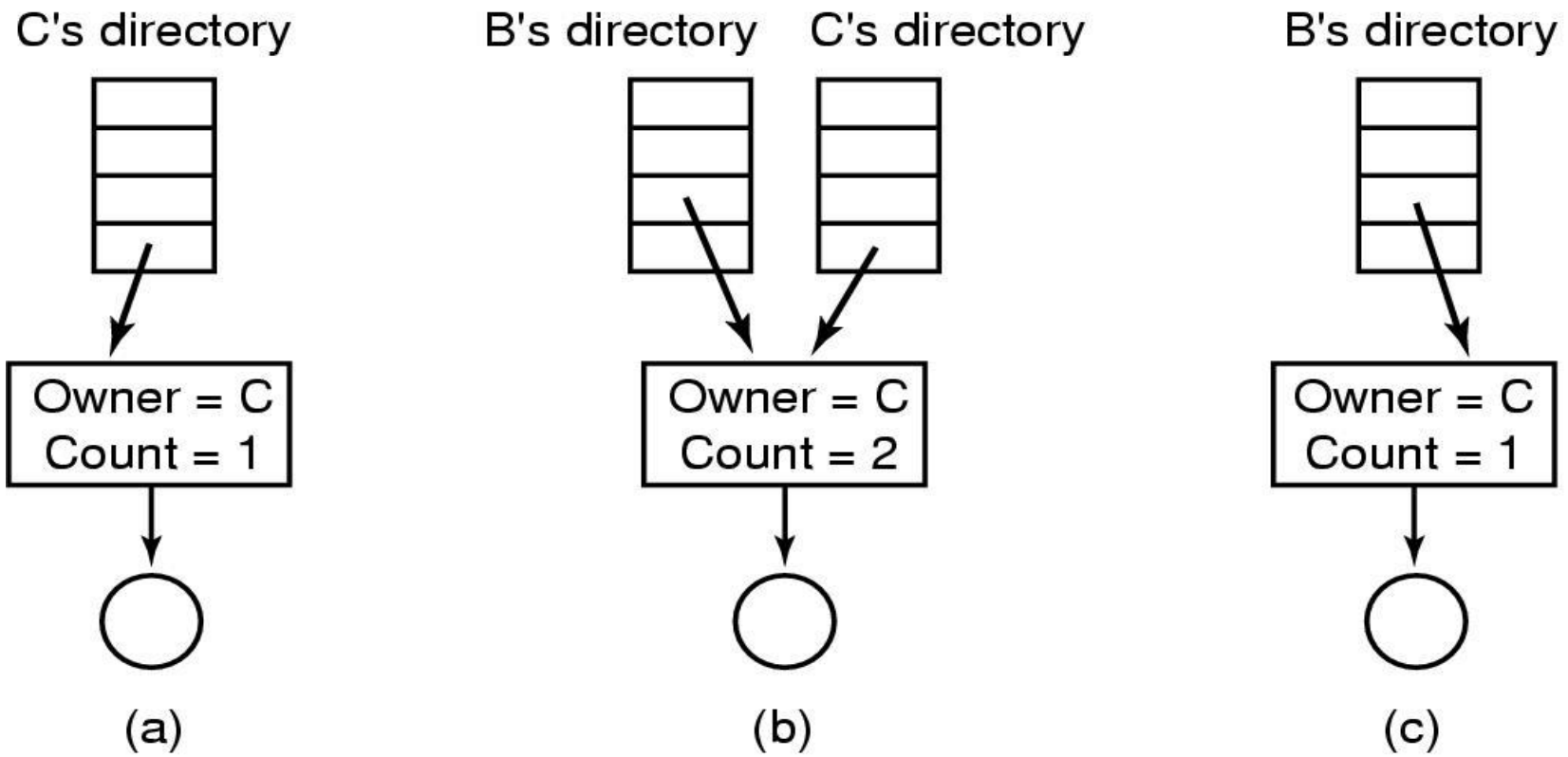
Directed Acyclic Graph (DAG) Directories



Directed Acyclic Graph Directories

- Have shared subdirectories and files.
- Entry may have two different names (aliasing).
- If *dict* deletes *list* $\Rightarrow$ dangling pointer – solutions:
  - Backpointers, so we can delete all pointers.
    Variable size records a problem.
  - Backpointers using a daisy chain organization.
  - Entry-hold-count solution.
- Newer type of directory entry:
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

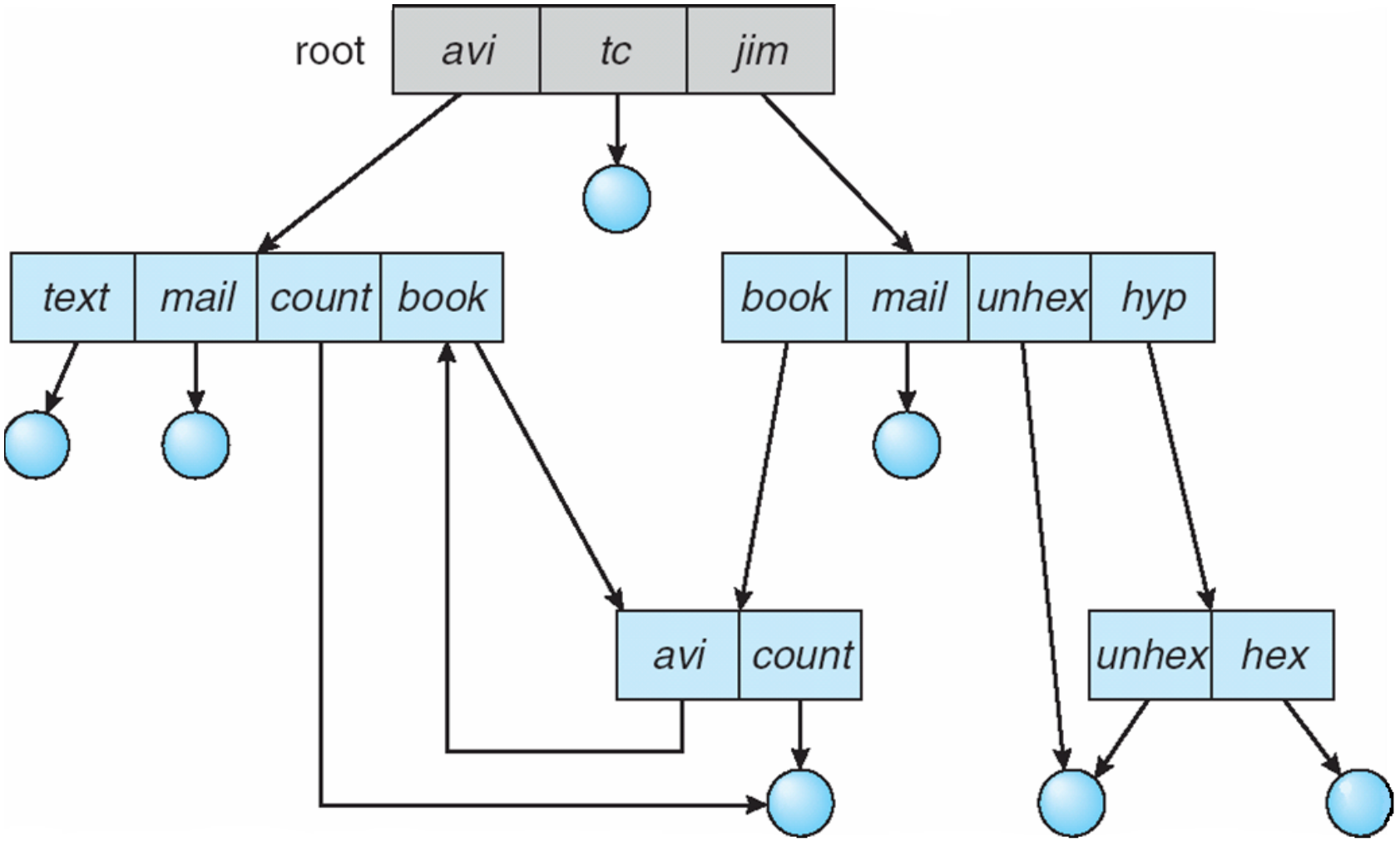
Shared File Example



(a) Situation prior to linking. (b) After the link is created.

(c) After the original owner removes the file.

General Graph Directory

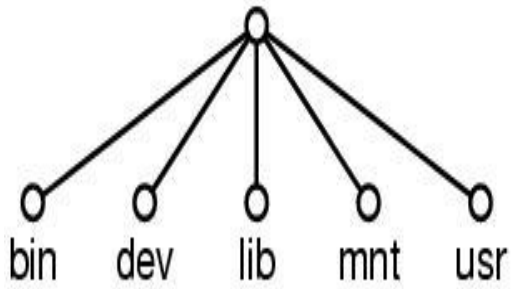


General Graph Directory

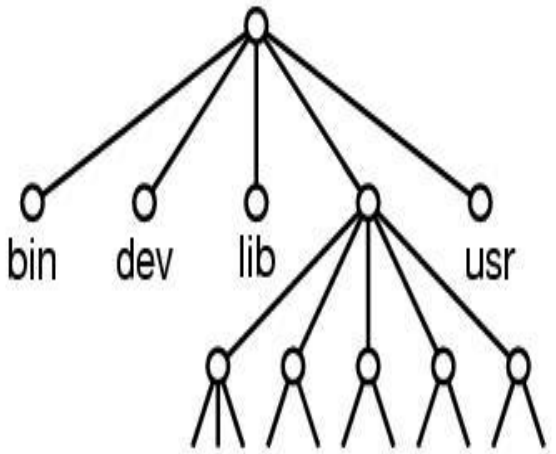
- How do we guarantee no cycles?
  - Allow only links to files, not subdirectories.
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK.
  - Use garbage collection.

File System Mounting

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

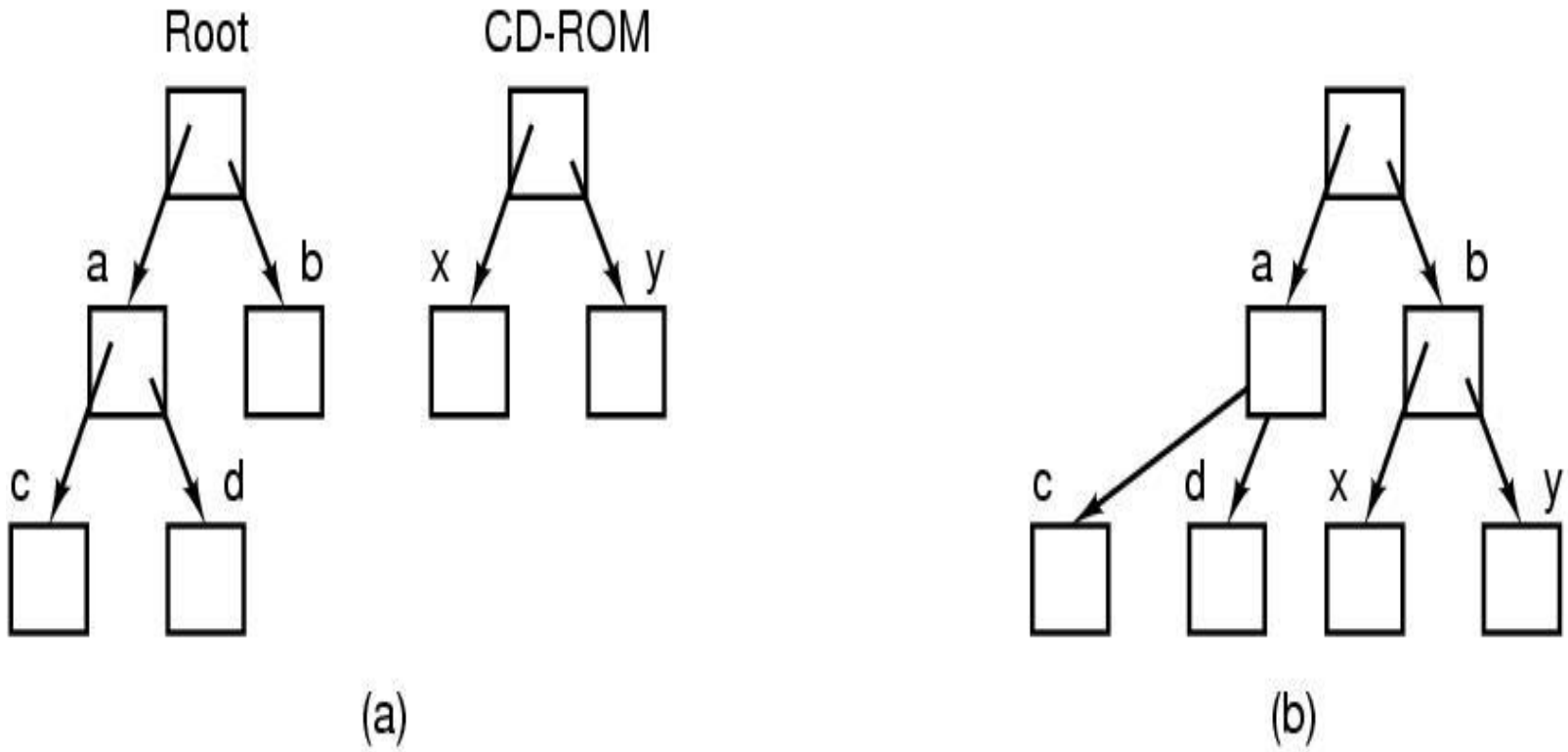


(a)



(b)

Example – File System Mounting



(a) Before mounting (b) Mounted system

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

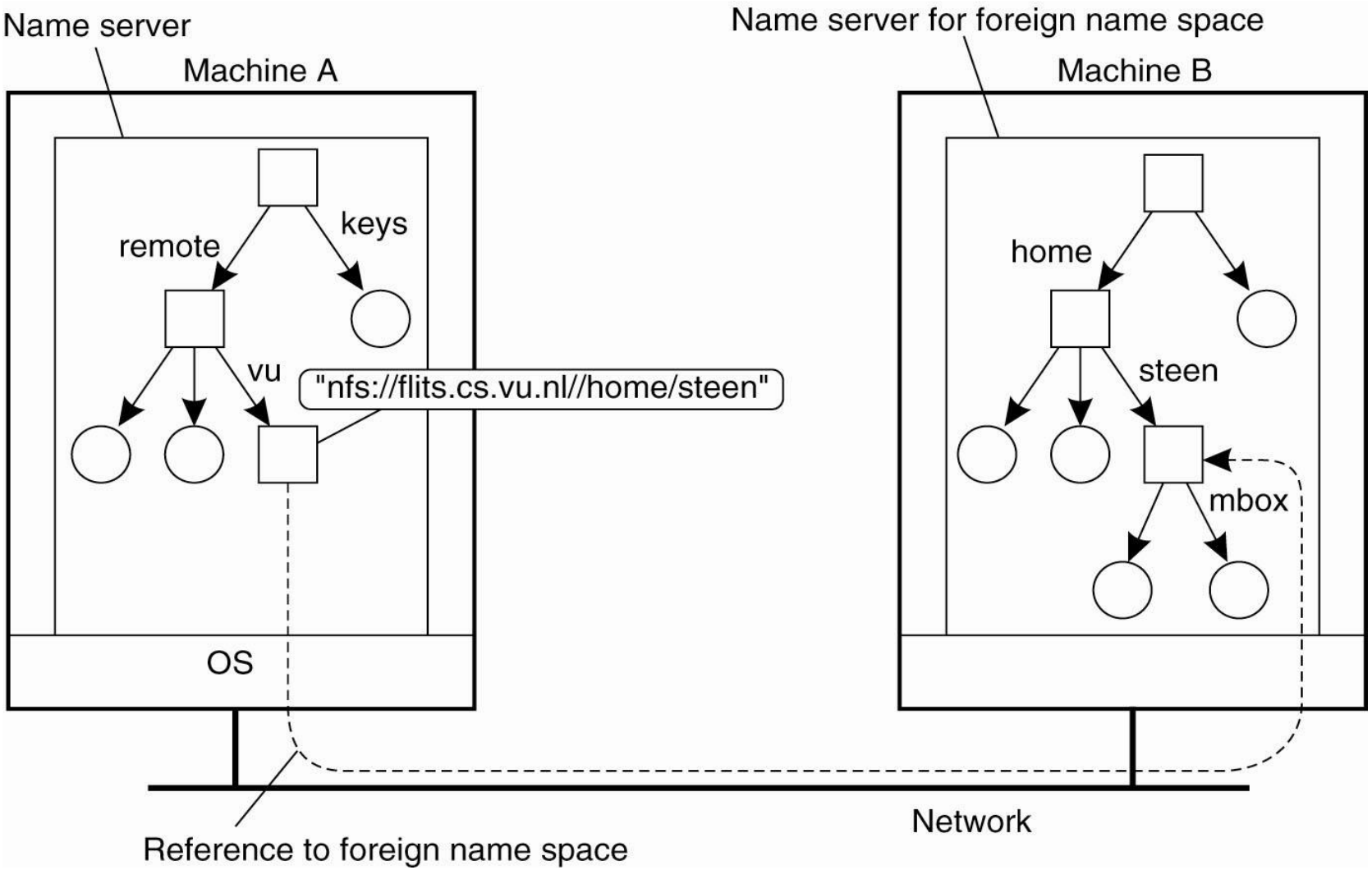
File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a *protection* scheme.
- On distributed systems, files may be shared across a network.
- Network File System (NFS) is a common distributed file-sharing method.
- If multi-user system:
  - **User IDs** identify users, allowing permissions and protections to be per-user.
  - **Group IDs** allow users to be in groups, permitting group access rights.
  - Owner/Group of a file/directory.

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems:
  - Manually via programs like FTP.
  - Automatically, seamlessly using **distributed file systems.**
  - Semi automatically via the **Web.**
- Client-server model allows clients to mount remote file systems from servers:
  - Server can serve multiple clients.
  - Client & user-on-client identification is insecure/complicated.
  - **NFS** is standard UNIX client-server file sharing protocol.
  - **CIFS** is standard Windows protocol.
  - Standard operating system file calls are translated into remote calls.
  - Distributed Information Systems (distributed naming services) such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing.

Mounting Remote Name Spaces



File Sharing – Failure Modes

- All file systems have failure modes:
  - For example corruption of directory structures or other non-user data, called metadata.
- Remote file systems add new failure modes, due to network failure, server failure.
- Recovery from failure can involve state information about status of each remote request.
- Stateless protocols such as NFS v3 include all information in each request, allowing easy recovery but less security.

Protection

- File owner/creator should be able to control:
  - what can be done
  - and by whom
- Types of access:
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

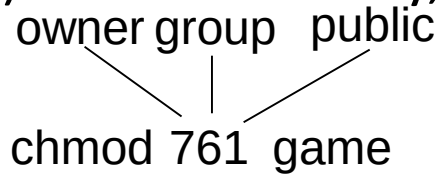
Access Lists and Groups

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

			RWX
a) **owner access**	7	⇒	1 1 1
			RWX
b) **group access**	6	⇒	1 1 0
			RWX
c) **public access**	1	⇒	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.
- Attach a group to a file

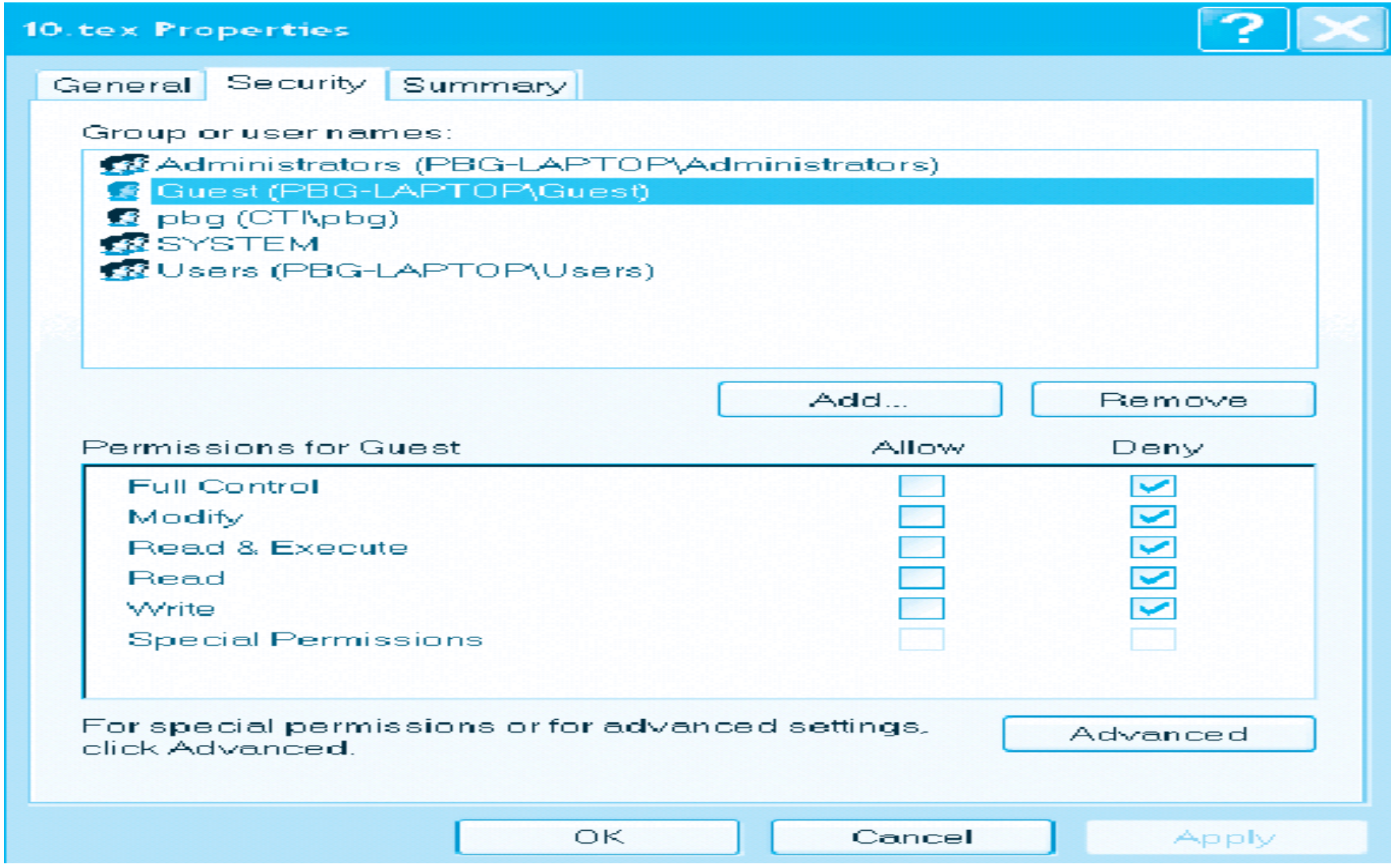
chgrp G game



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Windows XP
Access-control List Management



ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

File System Implementation

SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

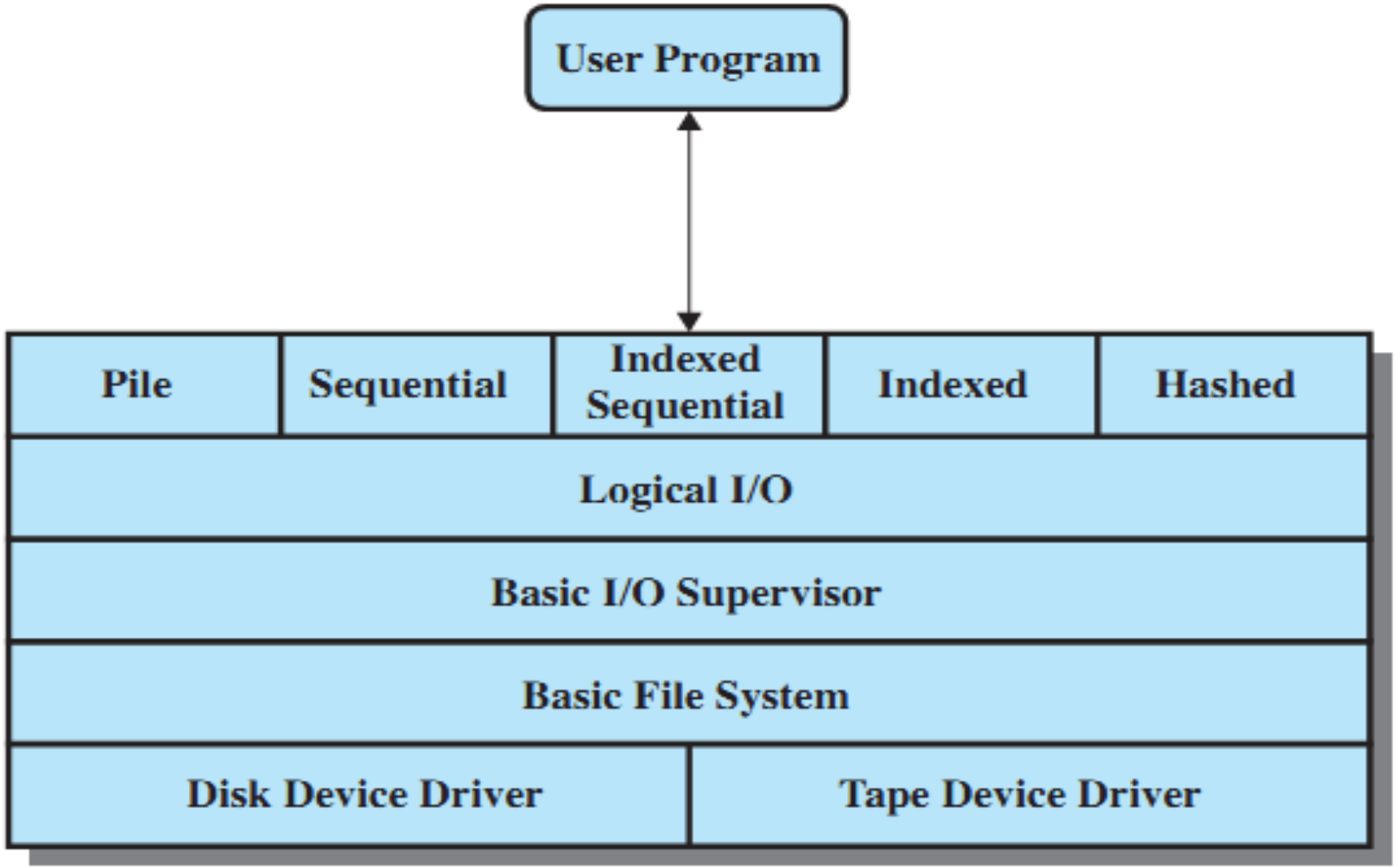
File System Implementation

- File-System Structure
- File-System Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery

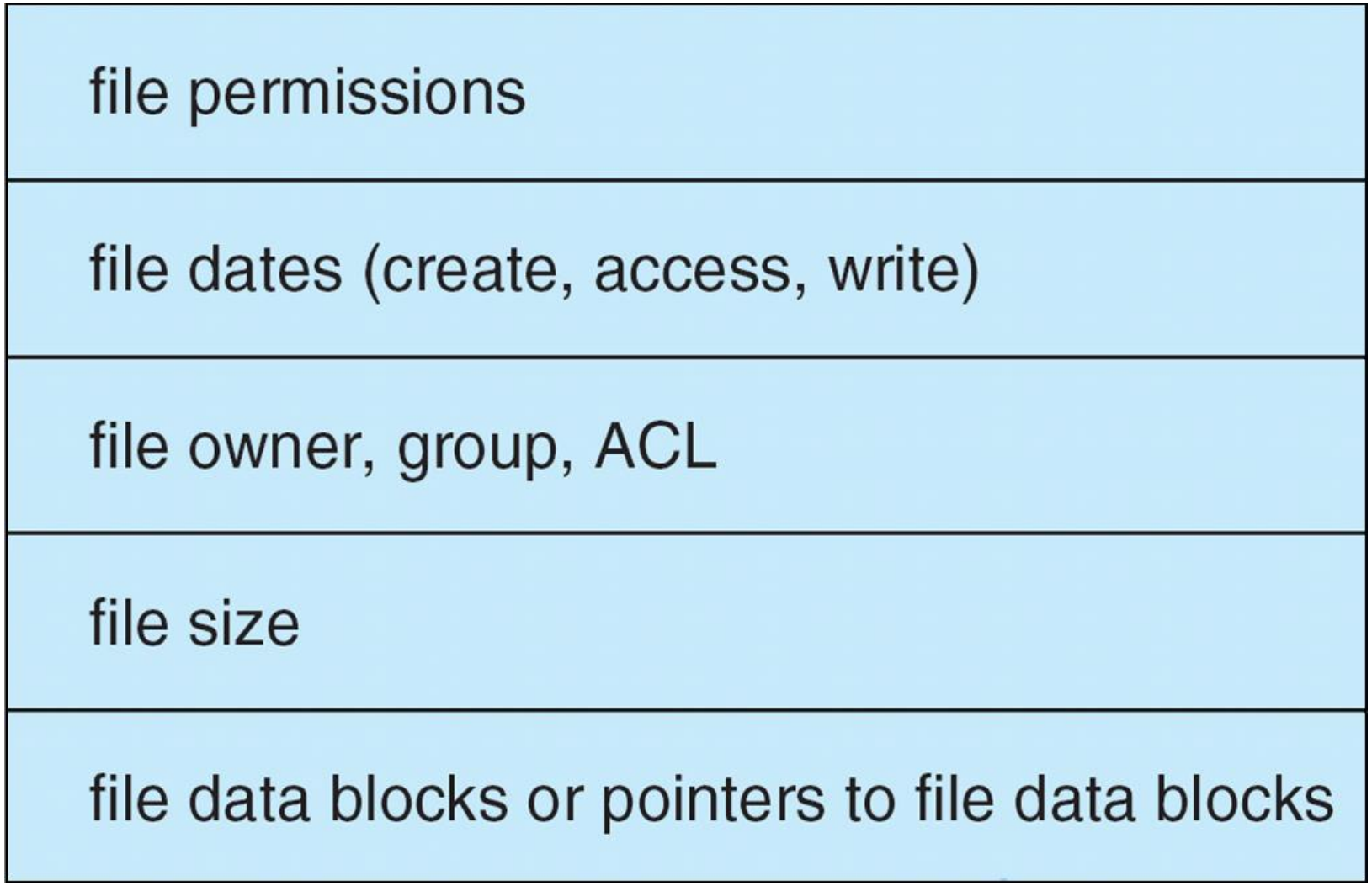
File-System Structure

- File structure:
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks):
  - Provided user interface to storage, mapping logical to physical.
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily.
- Disk provides in-place rewrite and random access:
  - I/O transfers performed in blocks of sectors (usually 512 bytes).
- File control block – storage structure consisting of information about a file.
- Device driver controls the physical device.
- File system organized into layers.

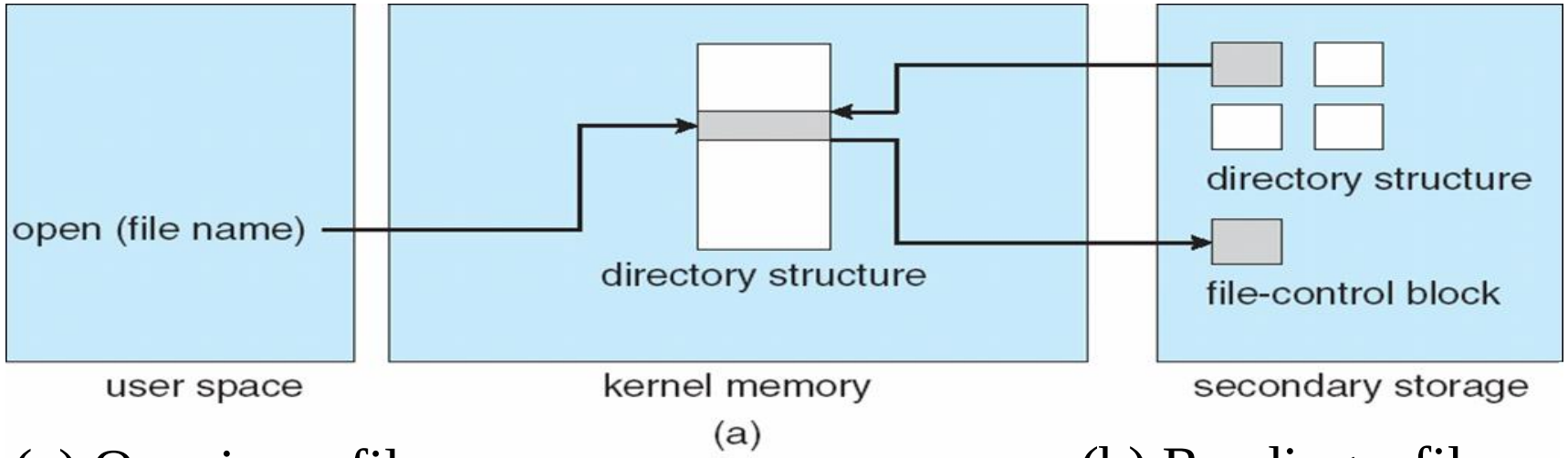
File System Software Architecture



A Typical File Control Block (FCB)

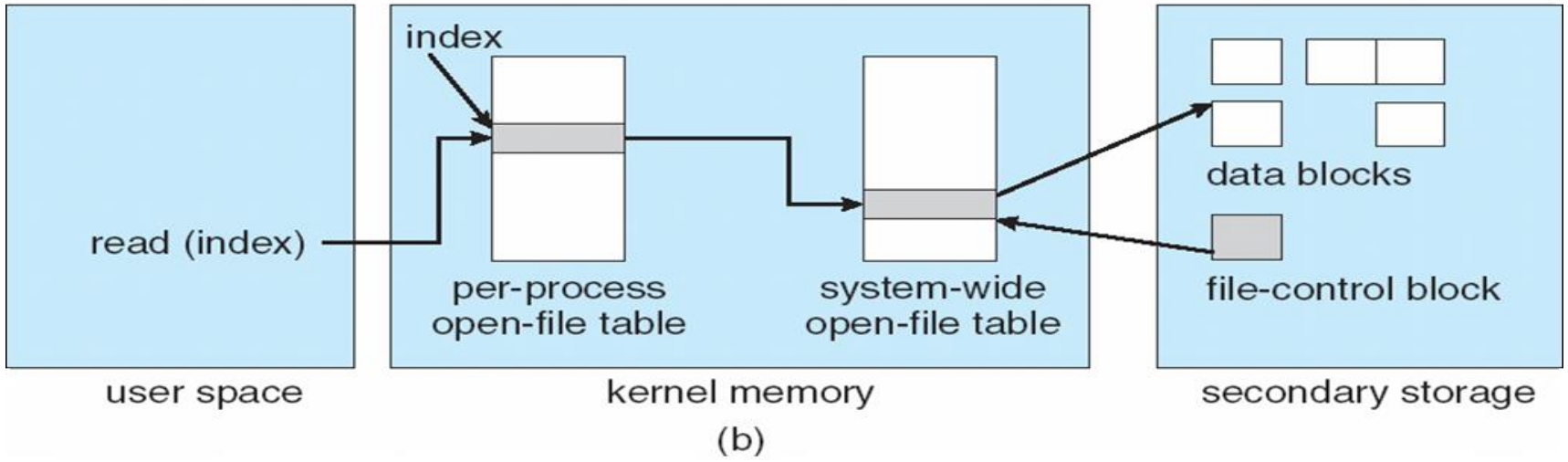


In-Memory File System Structures



(a) Opening a file

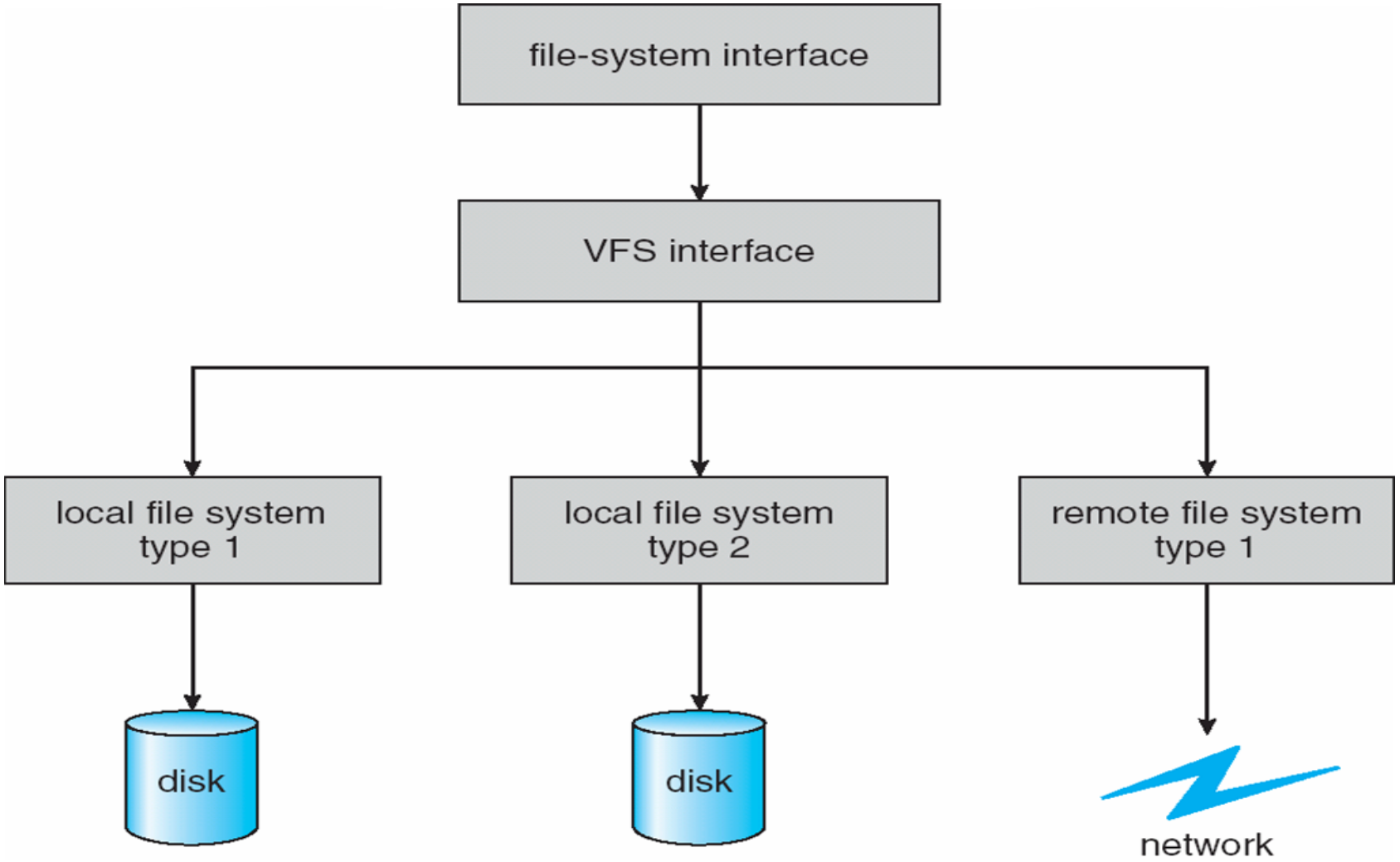
(b) Reading a file



Virtual File Systems

- 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.

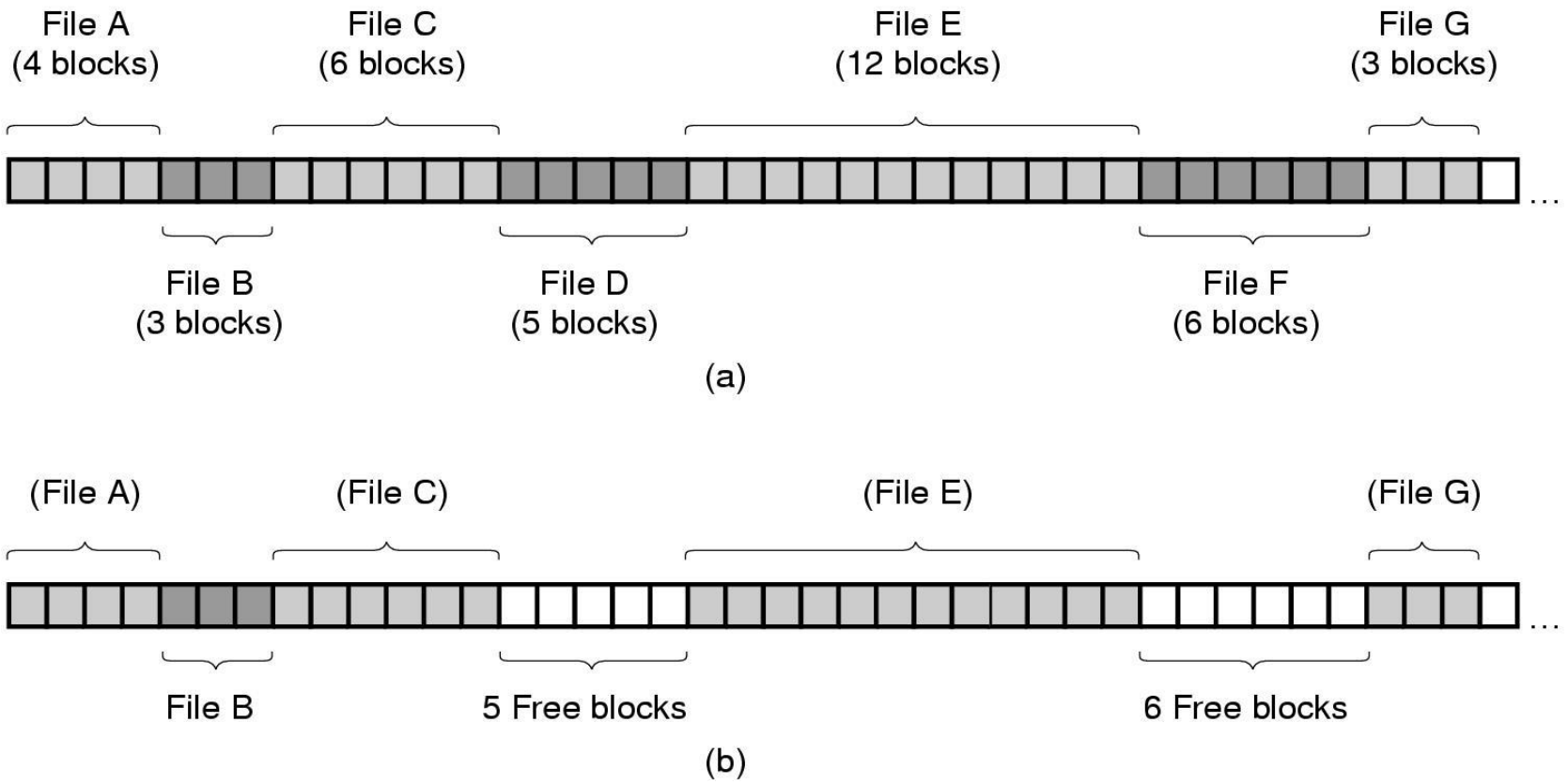
Schematic View of Virtual File System



Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
  1. Contiguous allocation
  2. Chained/Linked allocation
  3. Indexed allocation
  4. Combined schemes

Contiguous Allocation Example

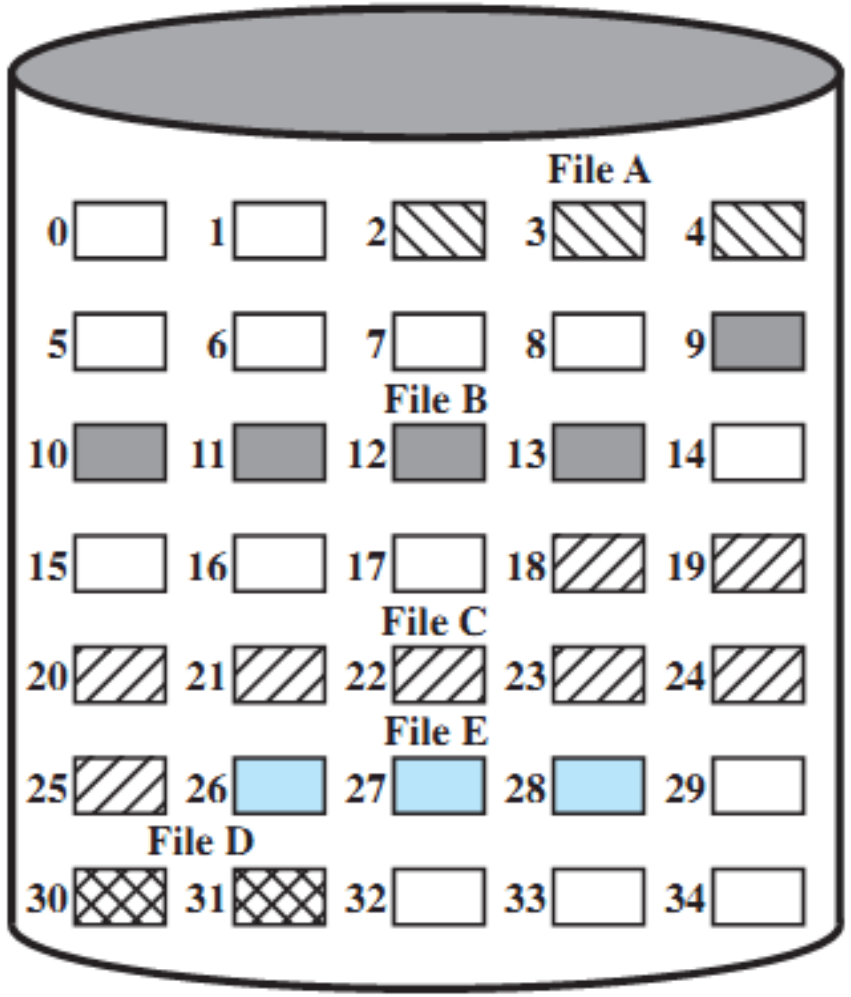


(a) Contiguous allocation of disk space for 7 files.
(b) The state of the disk after files D and F have been removed.

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple: only starting location (block #) and length (number of blocks) required.
- Enables random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

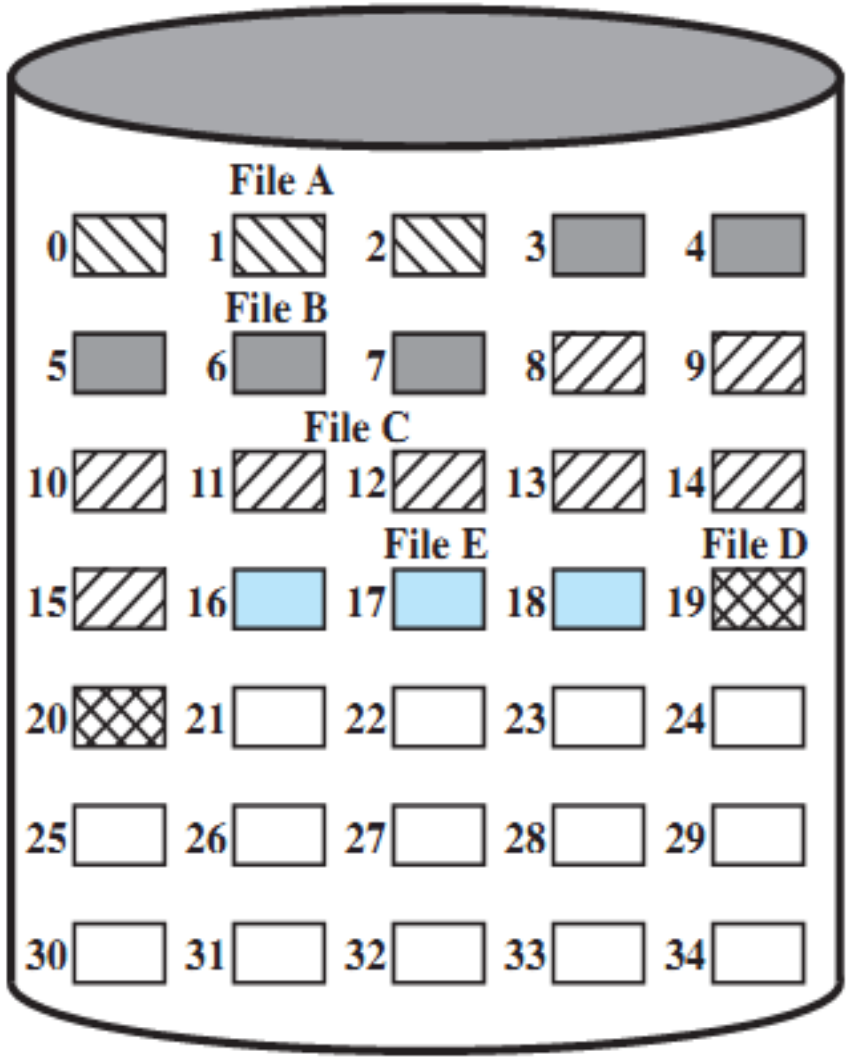
Contiguous File Allocation Example



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Contiguous file allocation (after compaction)



File Allocation Table

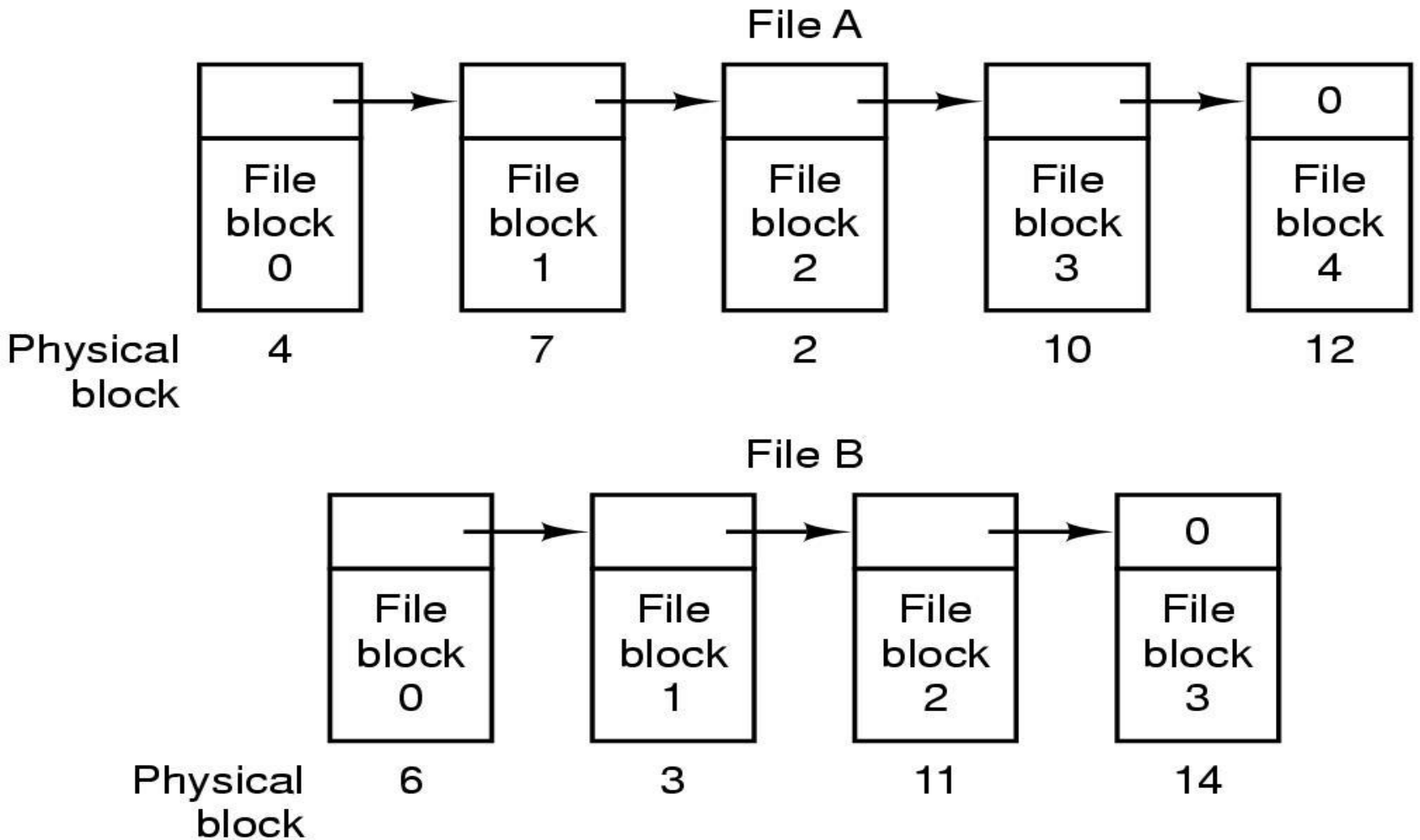
File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Extent-Based Systems

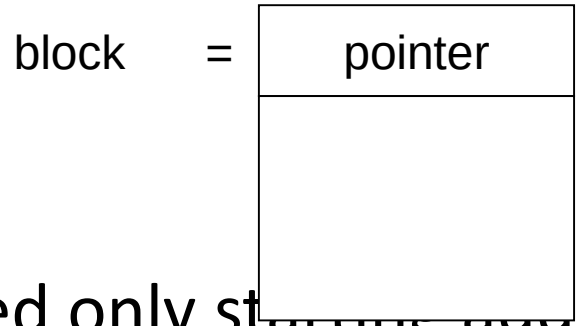
- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme.
- Extent-based file systems allocate disk blocks in extents.
- An extent is a contiguous block of disks:
  - Extents are allocated for file allocation.
  - A file consists of one or more extents.

Chained/Linked Allocation Example



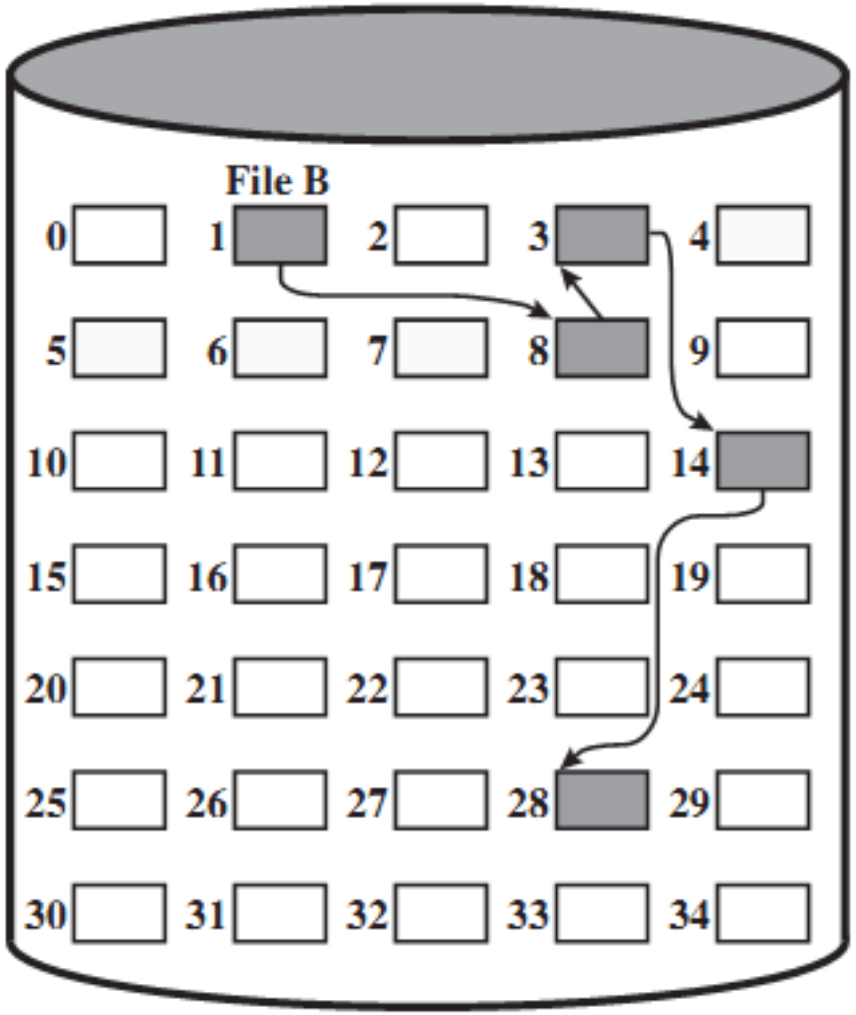
Chained Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



- Simple: need only starting address.
- Free-space management: no waste of space.
- No random access.

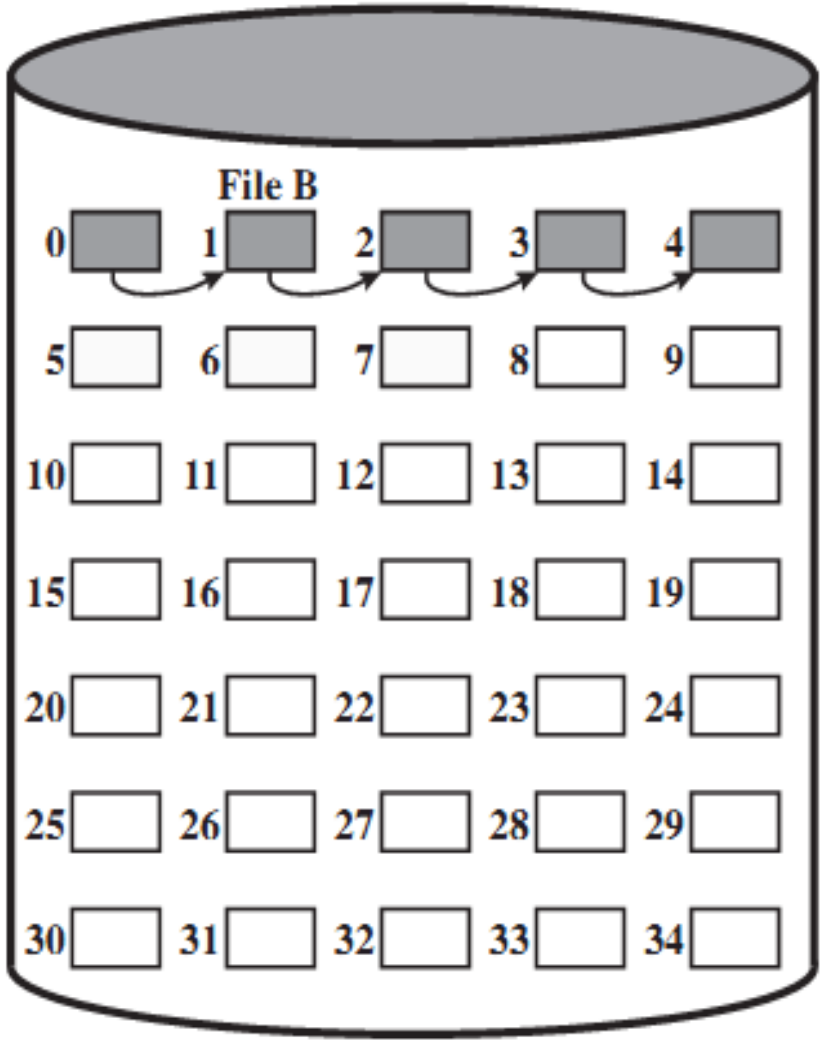
Chained File Allocation Example



File Allocation Table

File Name	Start Block	Length
• • •	• • •	• • •
File B	1	5
• • •	• • •	• • •

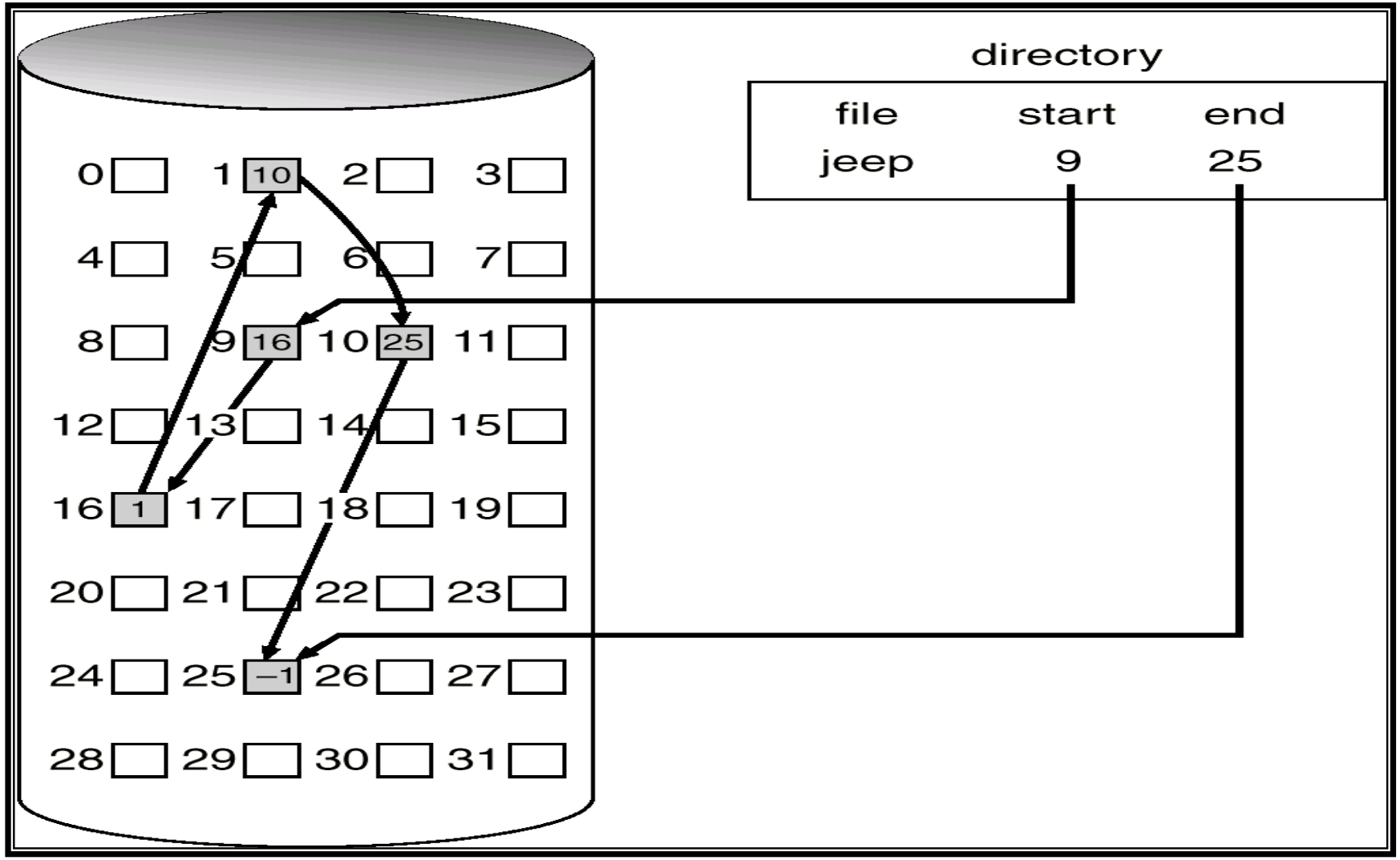
Chained File Allocation (after consolidation)



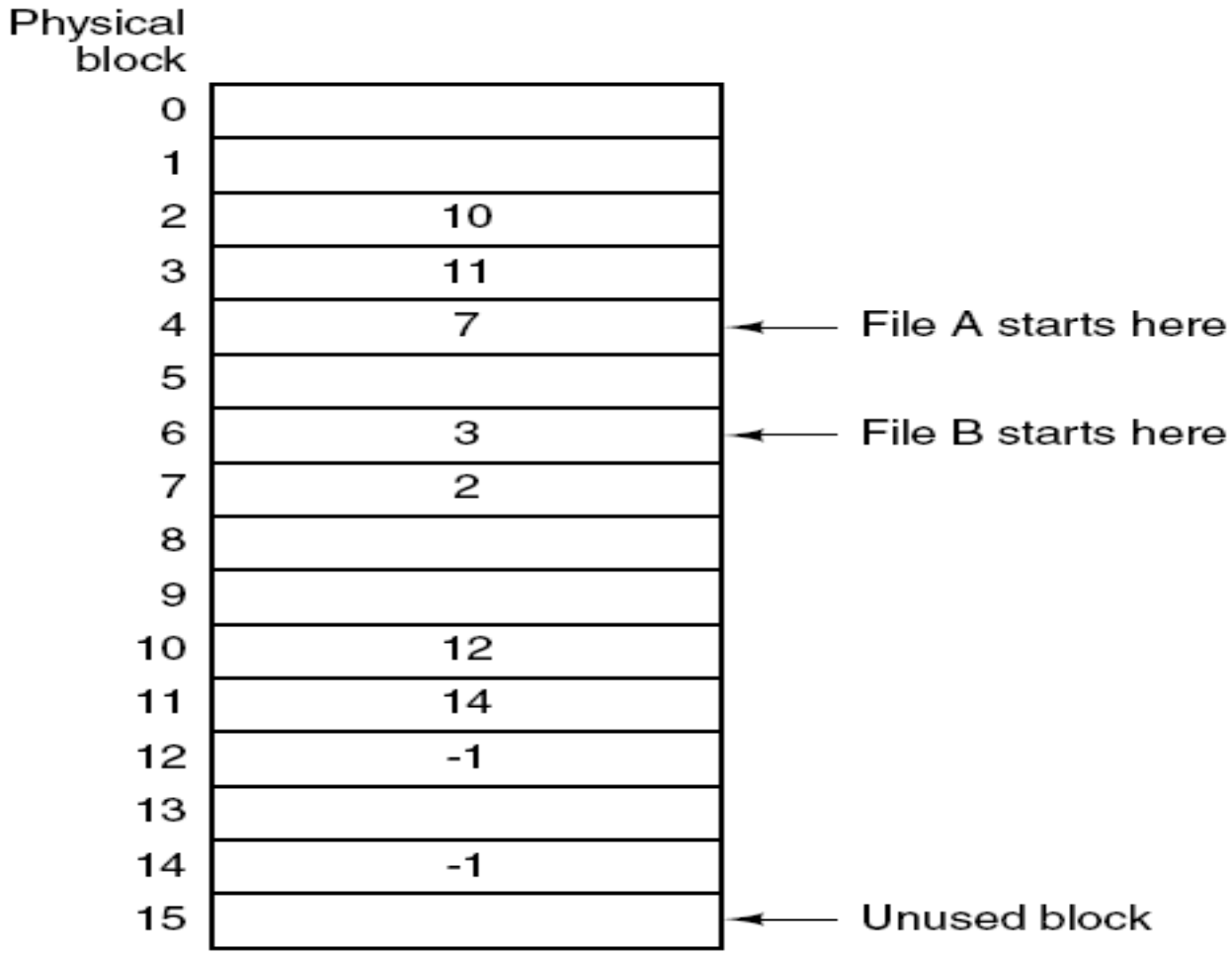
File Allocation Table

File Name	Start Block	Length
• • •	• • •	• • •
File B	0	5
• • •	• • •	• • •

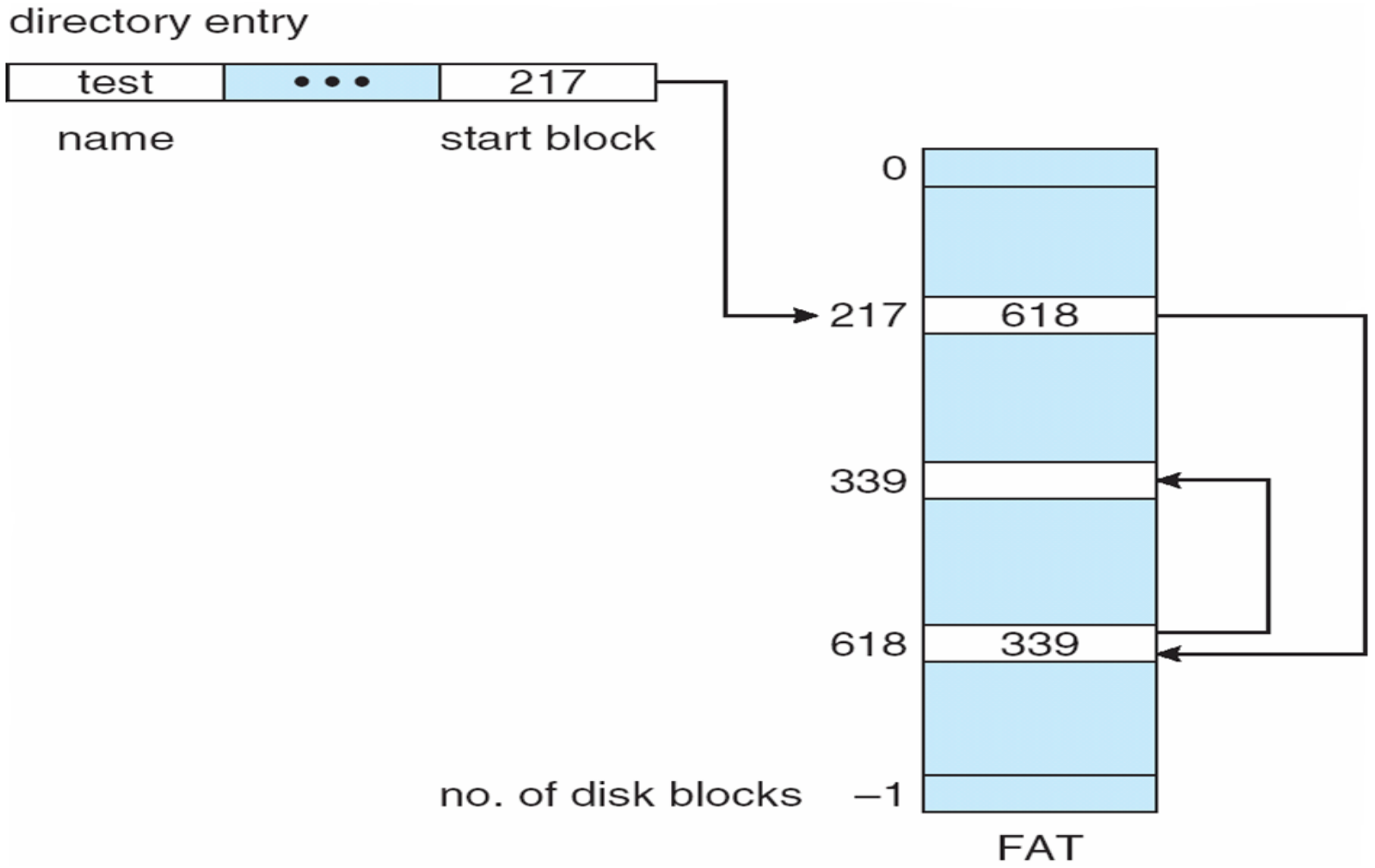
Another Chained Allocation Example



Linked List Allocation Using a Table in Memory

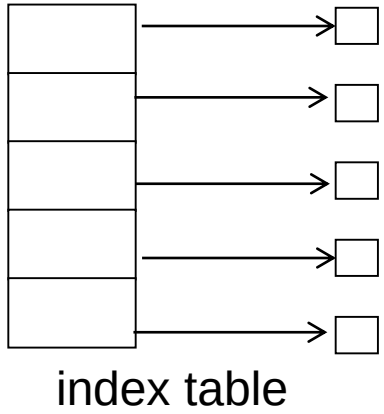


File-Allocation Table (FAT) Example

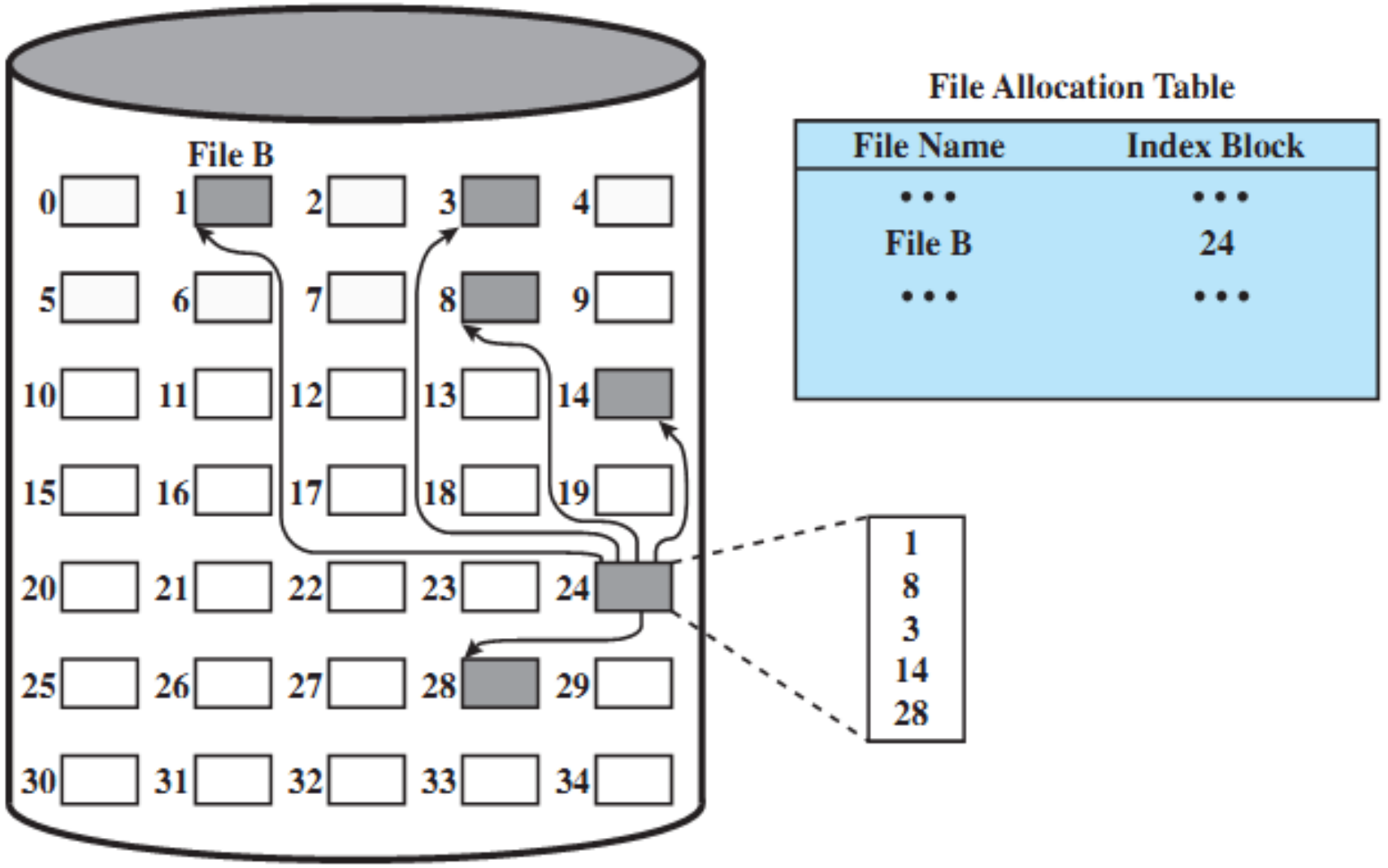


Indexed Allocation

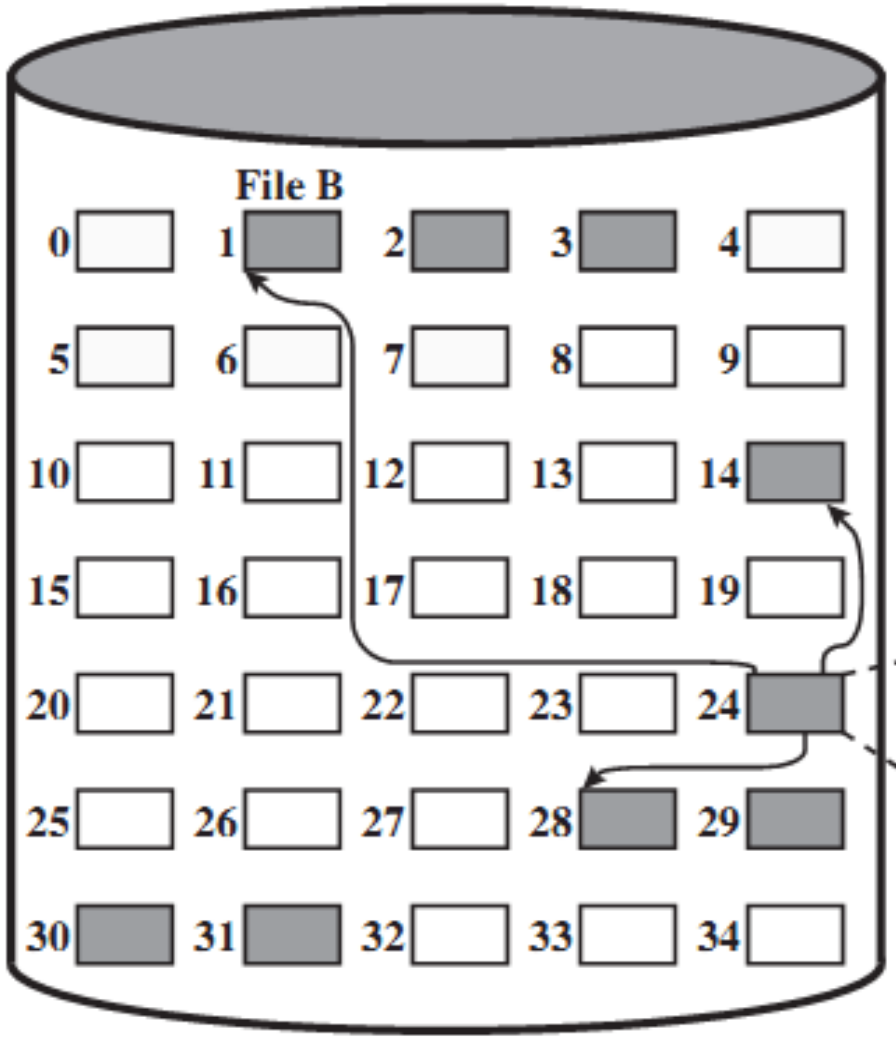
- Brings all pointers together into the *index block.*
- It’s a logical view.
- Need index table.
- Provides random access.
- Dynamic access without external fragmentation, but have overhead of index block.



Indexed File Allocation Example



Indexed File Allocation (variable-size)

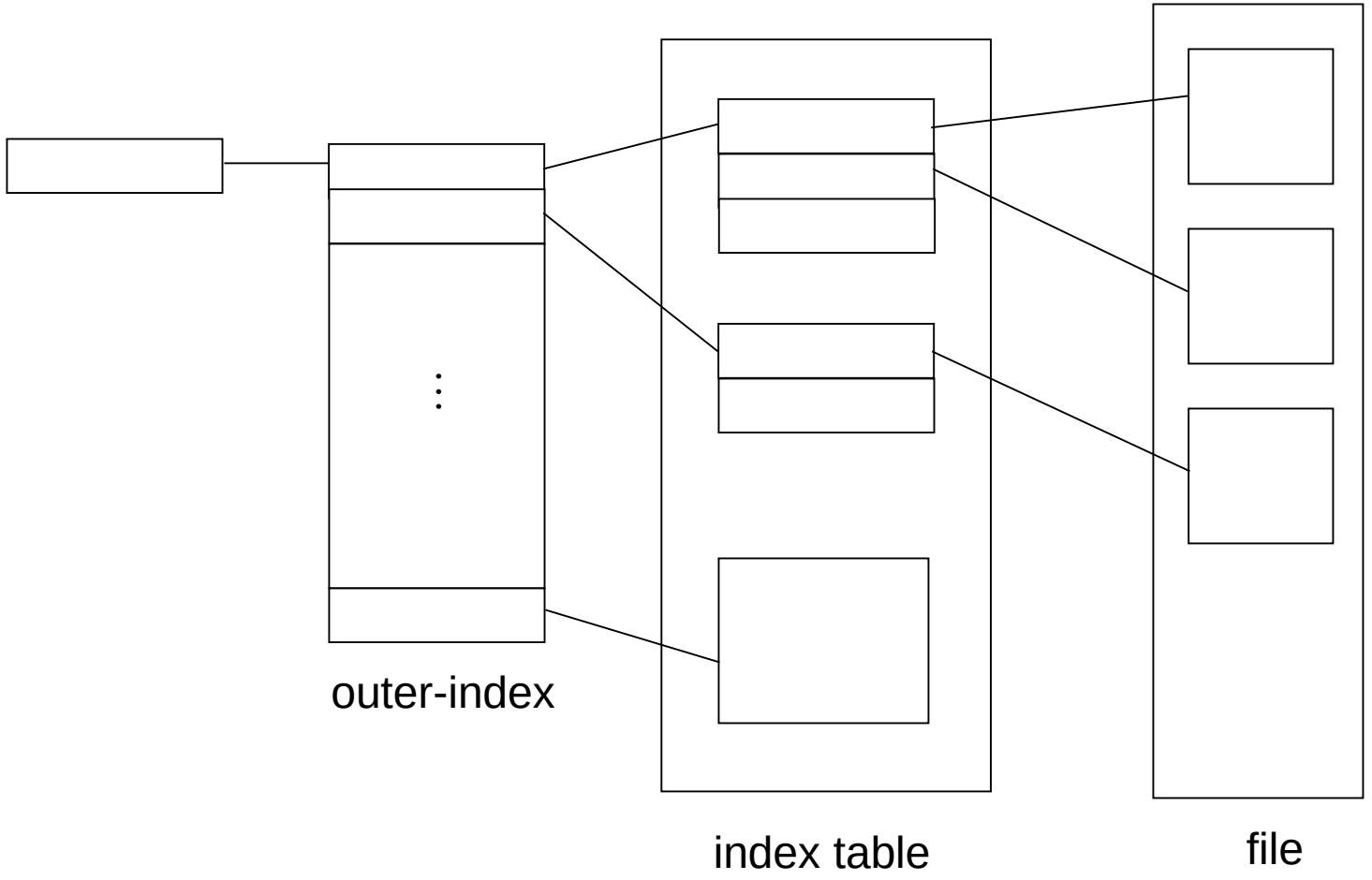


File Allocation Table

File Name	Index Block
• • •	• • •
File B	24
• • •	• • •

Start Block	Length
1	3
28	4
14	1

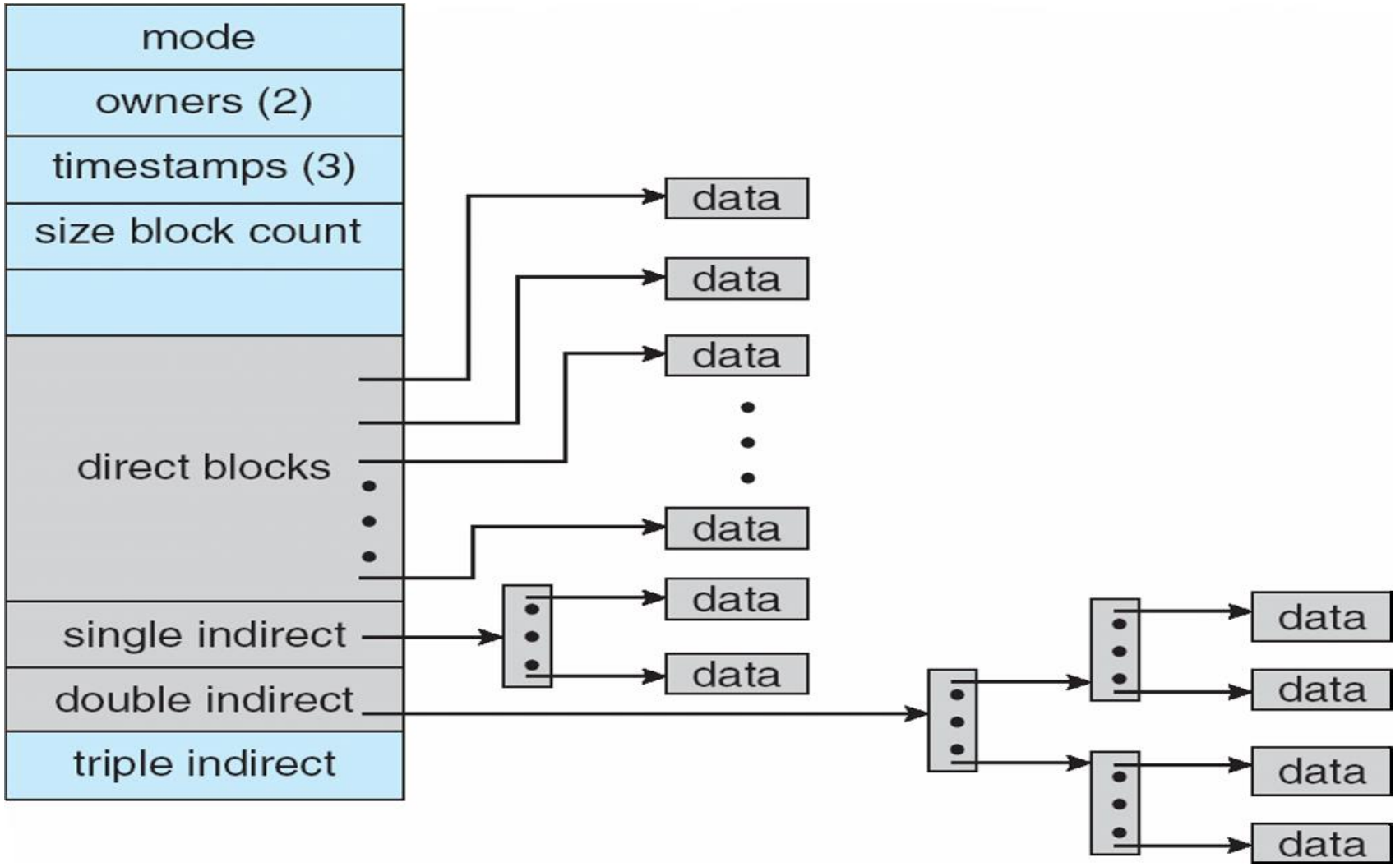
Indexed Allocation Mapping Example



Comparison of file allocation methods

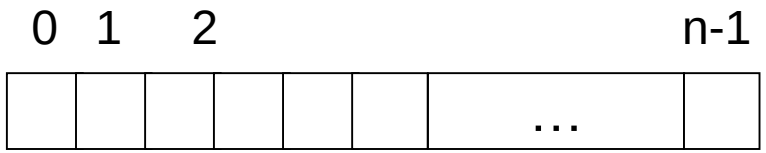
	Contiguous	Chained	Indexed	
Preallocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table size	One entry	One entry	Large	Medium

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



Free-Space Management (1)

- Bit vector (*n* blocks)



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

Block number calculation

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

Free-Space Management (2)

- Bit map requires extra space. For example:

  block size = 2^{12} bytes

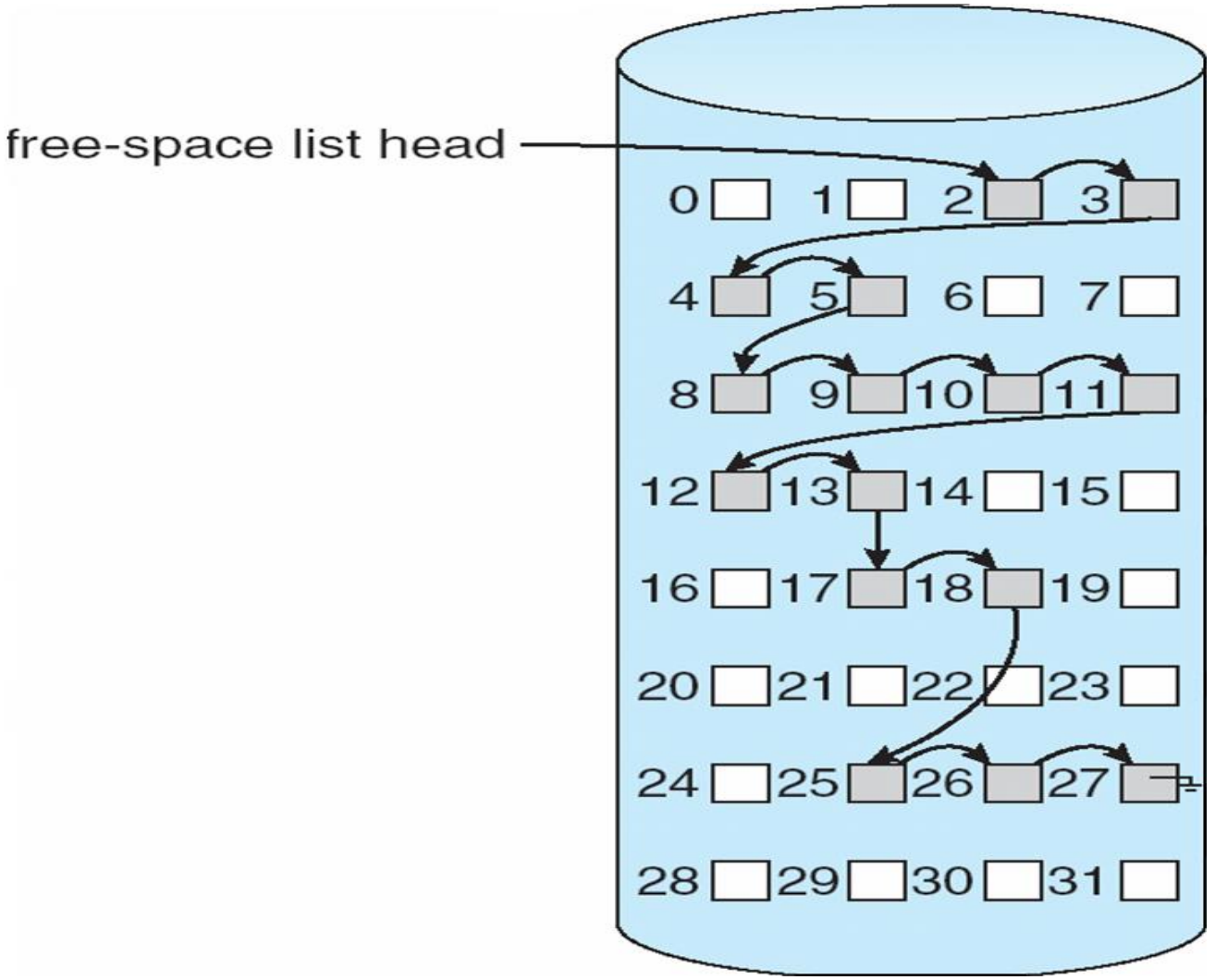
  disk size = 2^{30} bytes (1 gigabyte)

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

Free-Space Management (3)

- Need to protect:
  - Pointer to free list.
  - Bit map:
    - Must be kept on disk.
    - Copy in memory and disk may differ.
    - Cannot allow for block[*i*] to have a situation where bit[*i*] = 1 in memory and bit[*i*] = 0 on disk.
  - Solution:
    - Set bit[*i*] = 1 in disk.
    - Allocate block[*i*].
    - Set bit[*i*] = 1 in memory.

Linked Free Space List on Disk



Directory Implementation

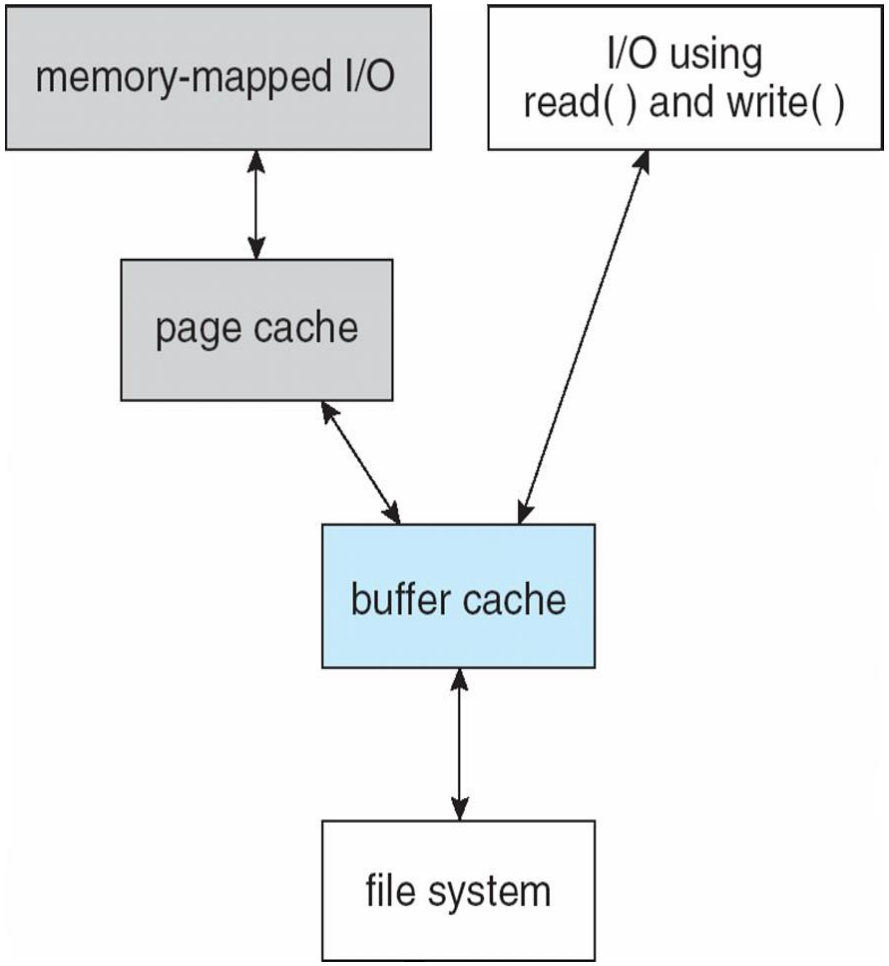
- Linear list of file names with pointer to the data blocks:
  - simple to program
  - time-consuming to execute
- Hash Table – linear list with hash data structure:
  - decreases directory search time
  - *collisions* – situations where two file names hash to the same location
  - fixed size

Efficiency and Performance

- Efficiency dependent on:
  - Disk allocation and directory algorithms.
  - Types of data kept in file's directory entry.
- Performance:
  - Disk cache: separate section of main memory for frequently used blocks.
  - Free-behind and read-ahead: techniques to optimize sequential access.
  - Improve PC performance by dedicating section of memory as virtual disk or RAM disk.

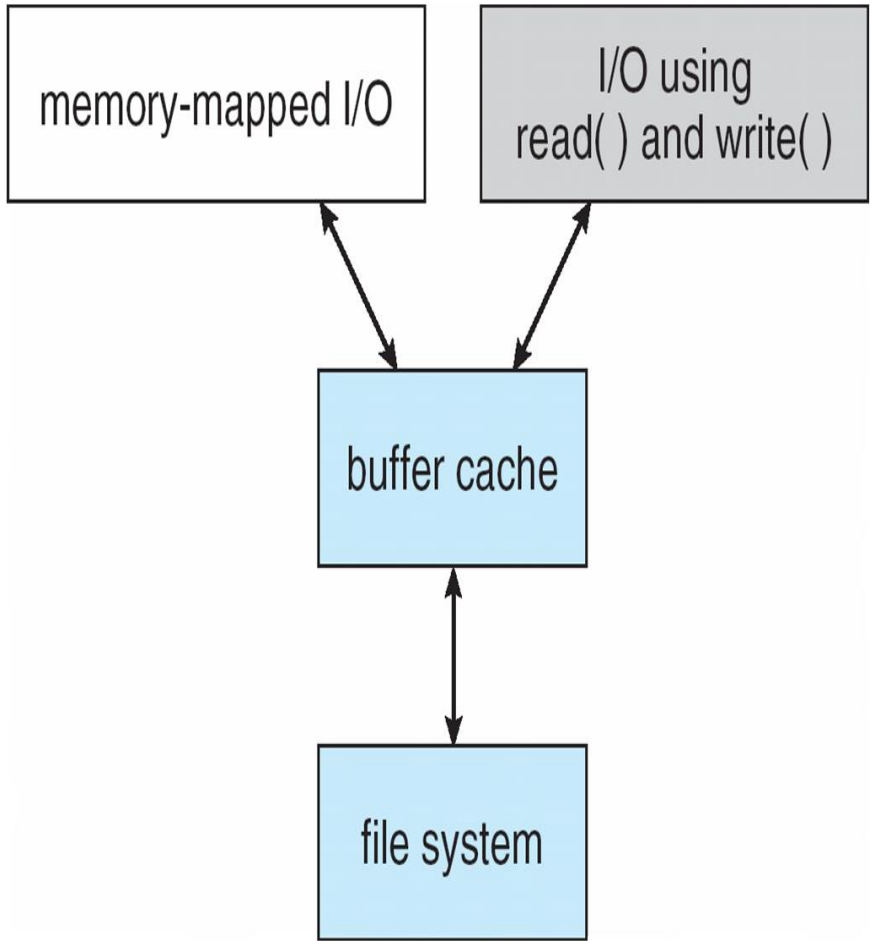
Page Cache

- 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.



Unified Buffer Cache

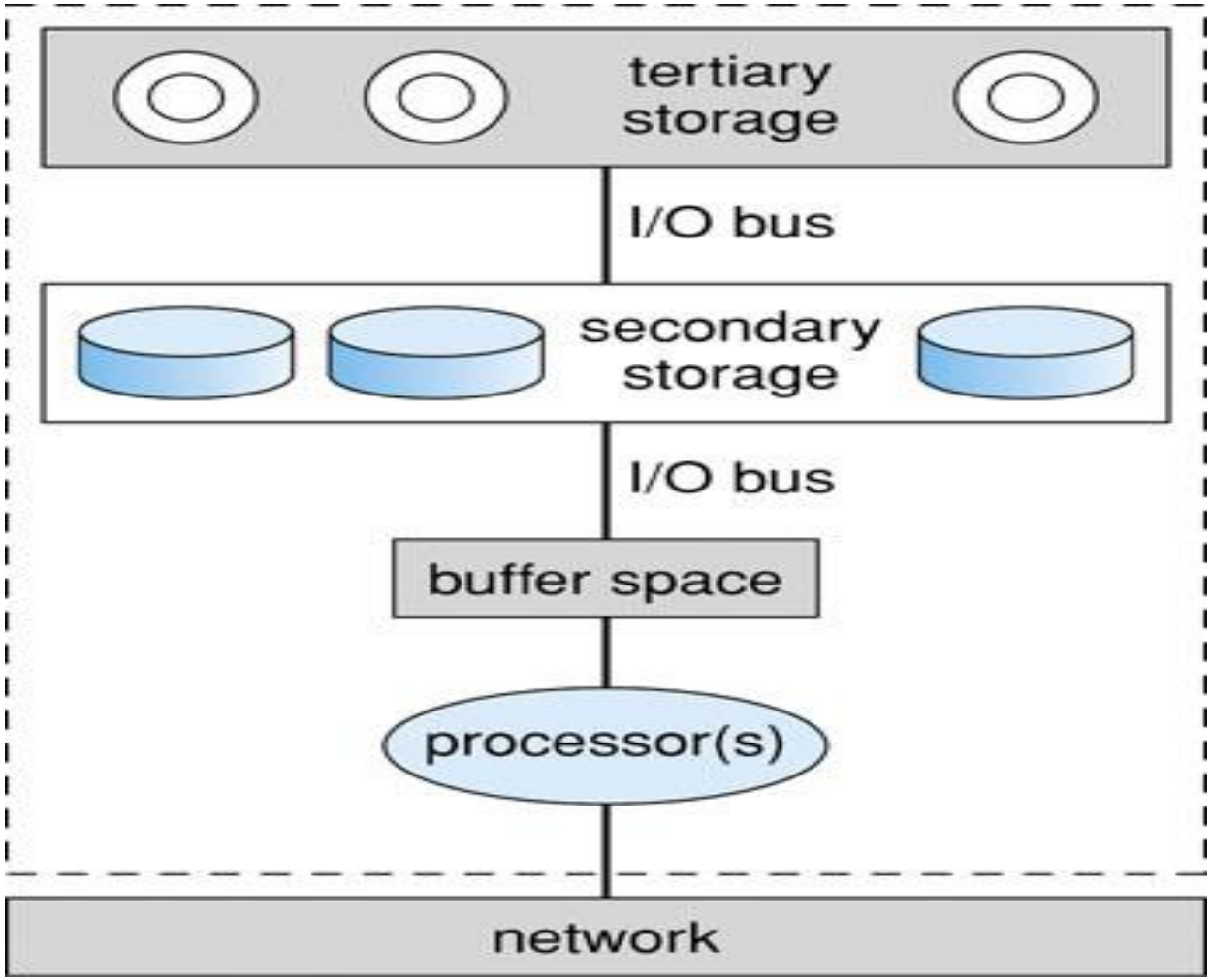
- A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O.



Recovery

- Consistency checking – compares data in directory structure with data blocks on disk,
  and tries to fix inconsistencies.
- Use system programs to back up data from disk to another storage device (floppy disk, magnetic tape).
- Recover lost file or disk by restoring data from backup.

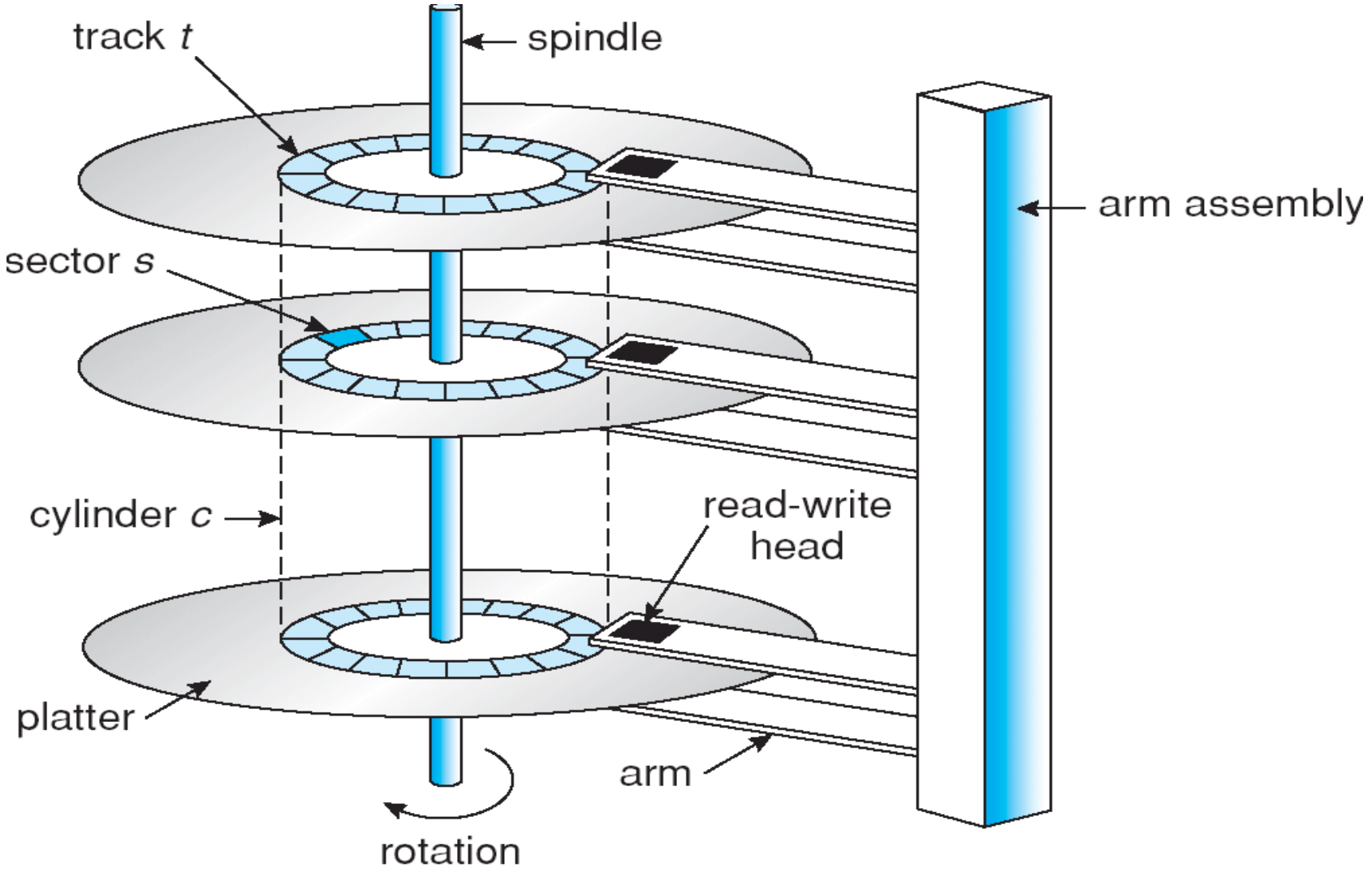
Storage Backup



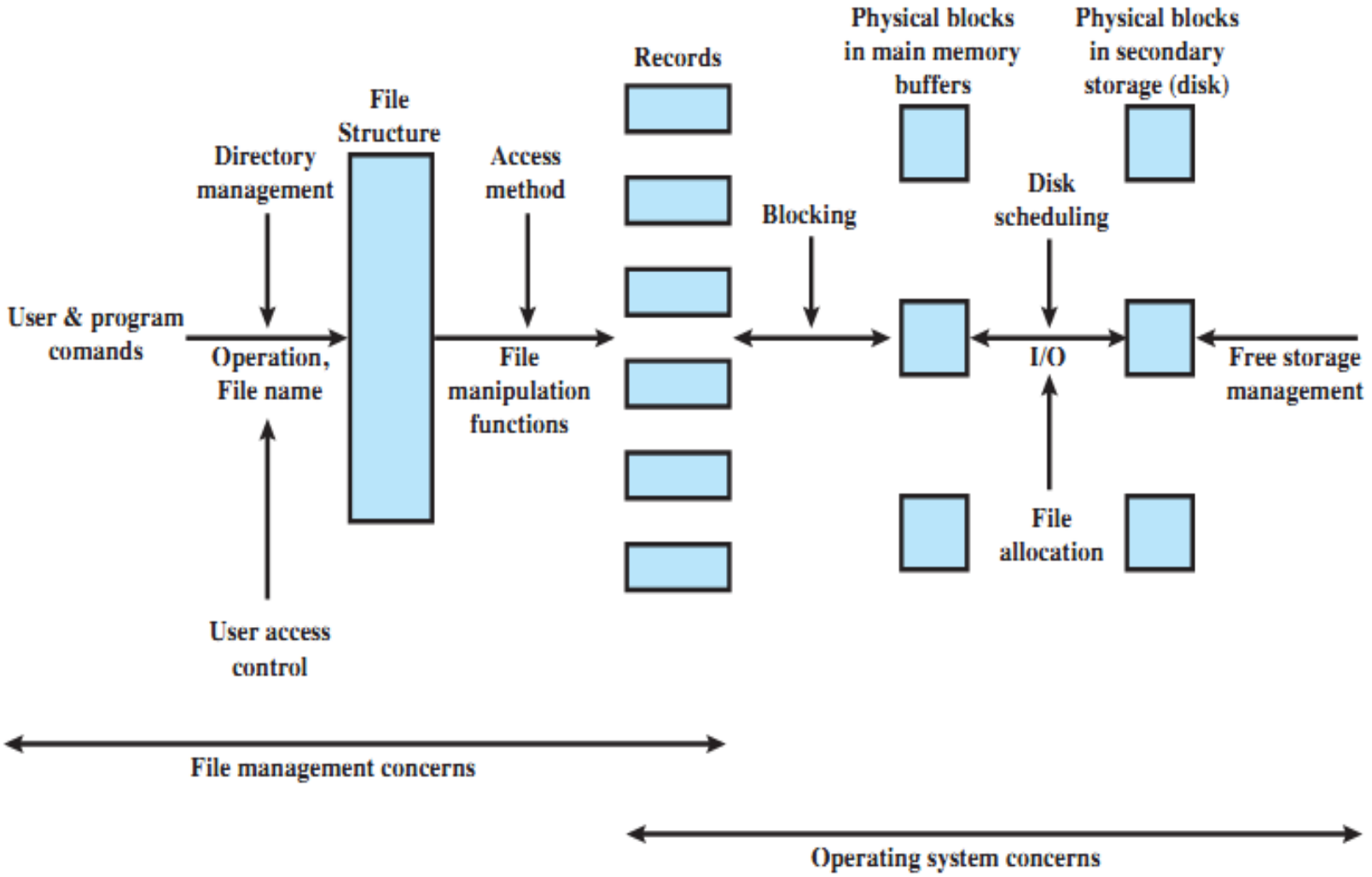
ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

Disk Scheduling

Moving-head Disk Mechanism



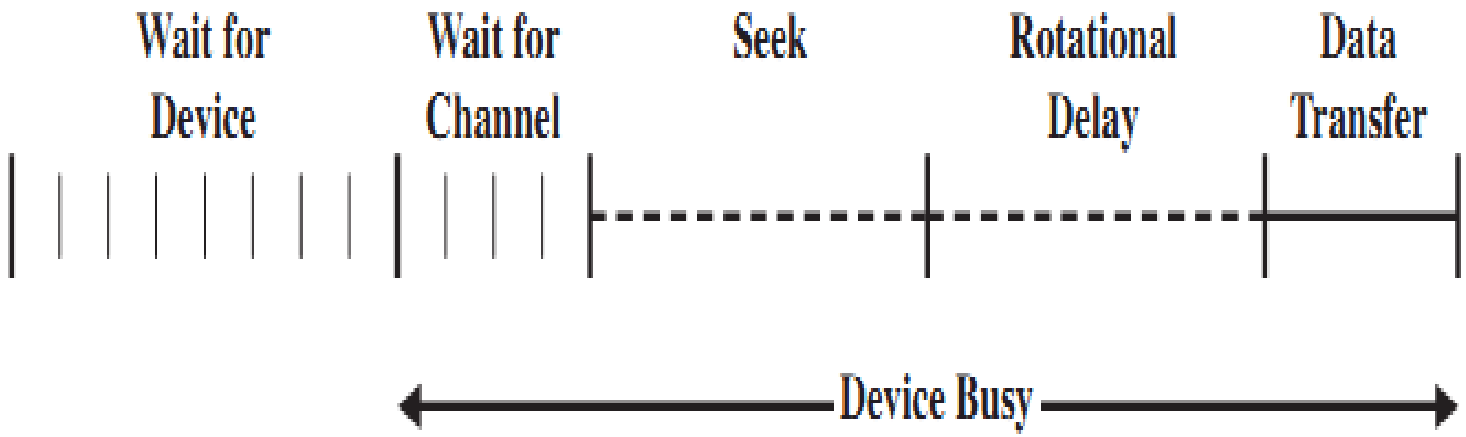
Elements of File Management



Disk Scheduling (1)

- 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 ≈ 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 last transfer.

Components of Disk I/O Transfer



ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Disk Scheduling (2)

- There are many sources of disk I/O request:
  - OS
  - System processes
  - Users processes
- I/O request includes input/output mode, disk address, memory address, number of sectors to transfer.
- OS maintains queue of requests, per disk or device.
- Idle disk can immediately work on I/O request, busy disk means work must queue:
  - Optimization algorithms only make sense when a queue exists.

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.

Disk Scheduling Algorithms

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”).
- Several algorithms exist to schedule the servicing of disk I/O requests.
- The analysis is true for one or many platters.
- We illustrate them with a I/O request queue (cylinders are between 0-199):

  queue = 98, 183, 37, 122, 14, 124, 65, 67

  head starts at 53

First Come First Serve (FCFS) Example

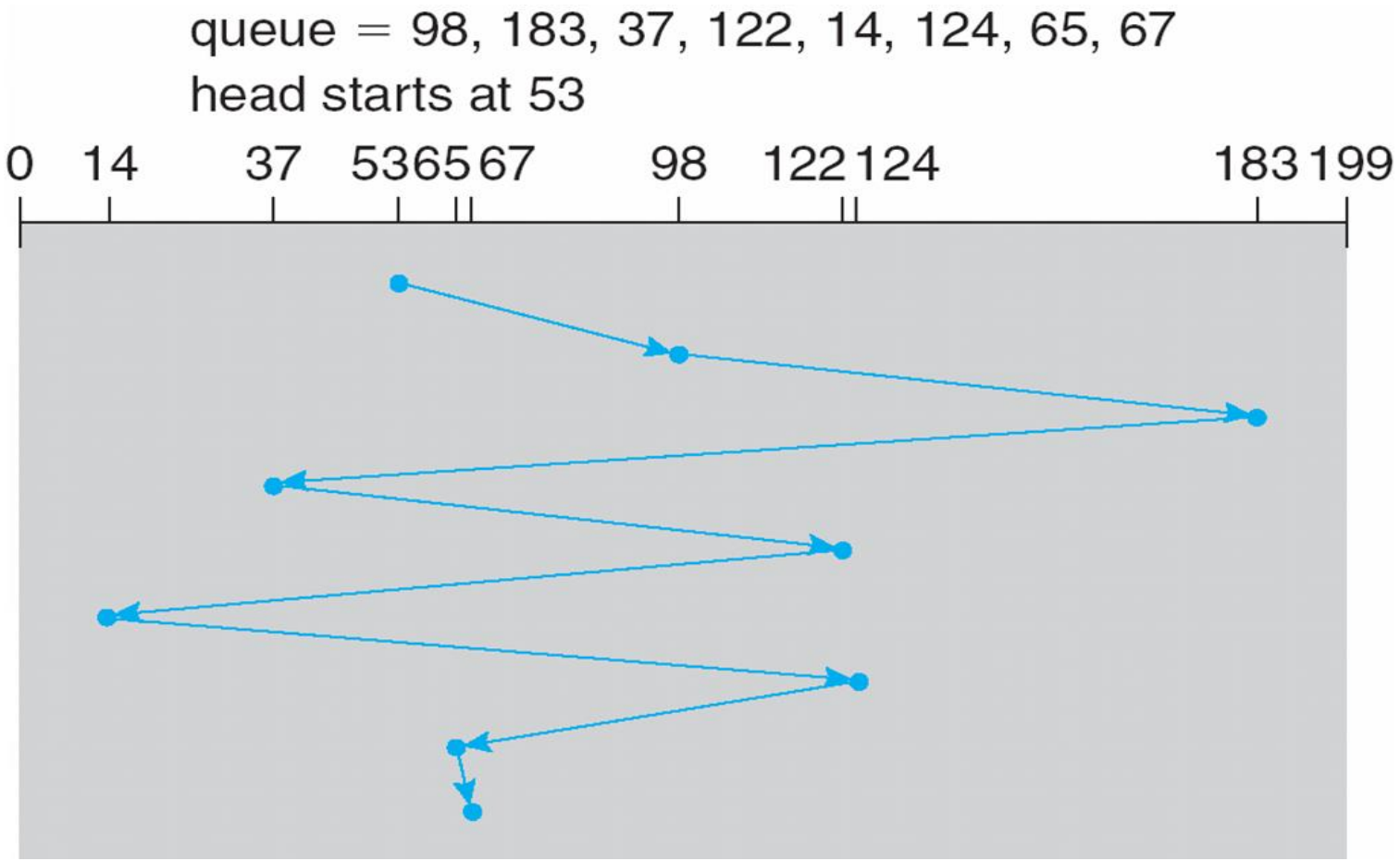


Illustration shows total head movement of 640 cylinders.

First Come First Serve (FCFS)

- Handle I/O requests sequentially.
- Fair to all processes.
- Approaches random scheduling in performance if there are many processes/requests.
- Suffers from global zigzag effect.

Shortest Seek Time First (SSTF) Example

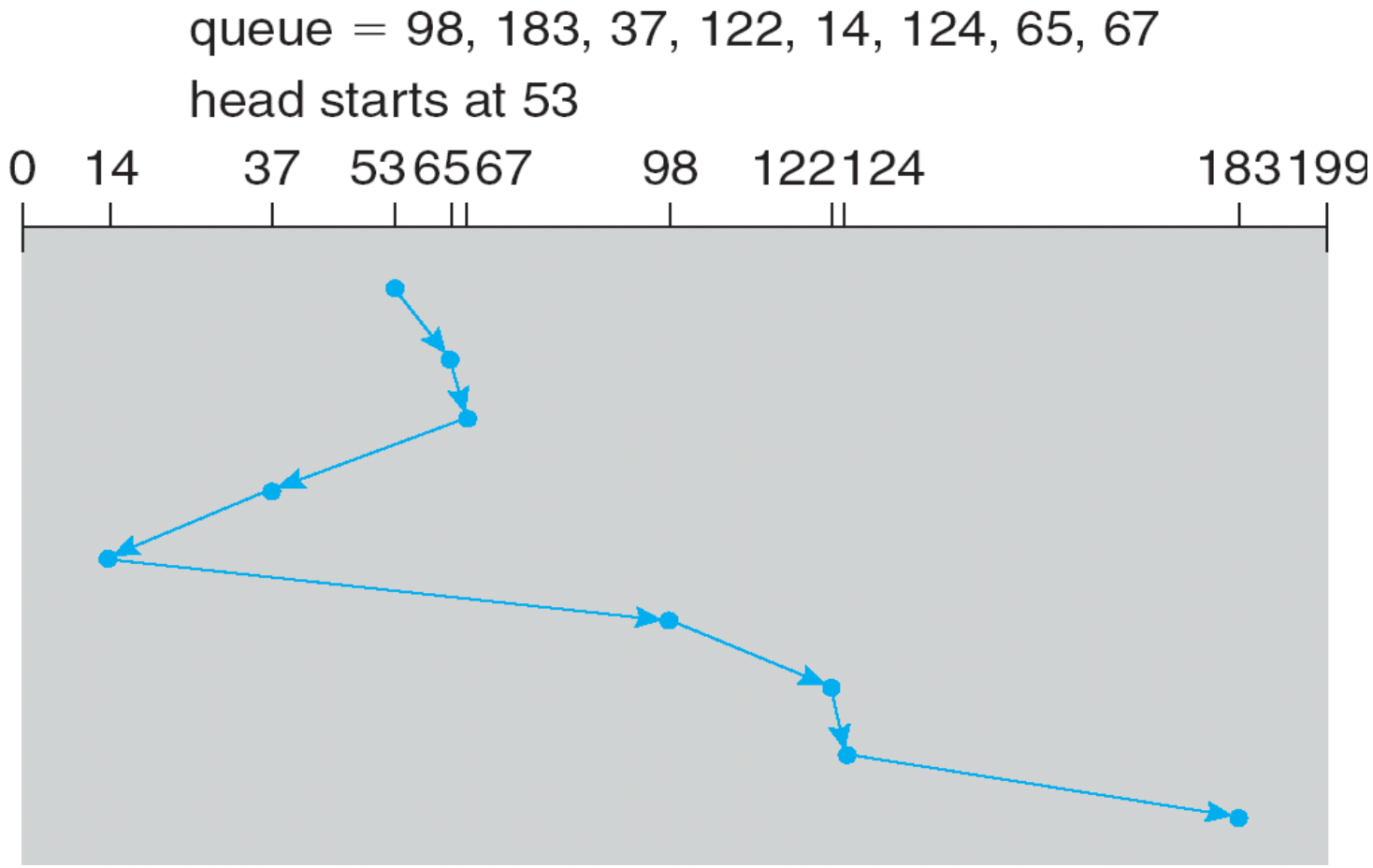


Illustration shows total head movement of 236 cylinders.

Shortest Seek Time First (SSTF)

- Selects the request with the minimum seek time from the current head position.
- Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances.
- It's biased in favor of the middle cylinders requests.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

Elevator Algorithms

- Algorithms based on the common elevator principle.
- Four combinations of Elevator algorithms:
  - Service in both directions or in only one direction.
  - Go until last cylinder or until last I/O request.

Direction \ Go until	Go until the last cylinder	Go until the last request
Service both directions	**Scan**	**Look**
Service in only one direction	**C-Scan**	**C-Look**

Scan Example

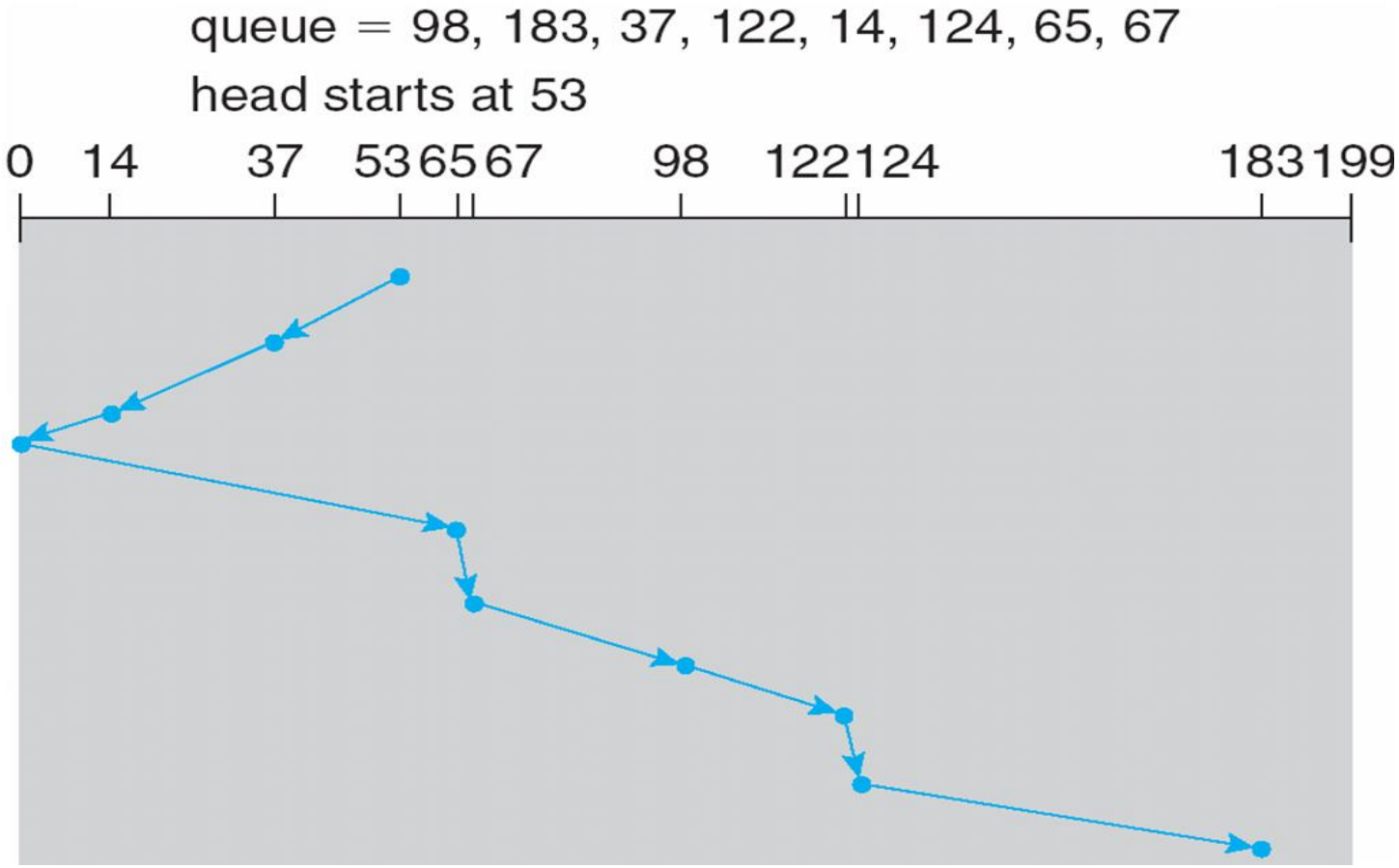


Illustration shows total head movement of 208 cylinders.

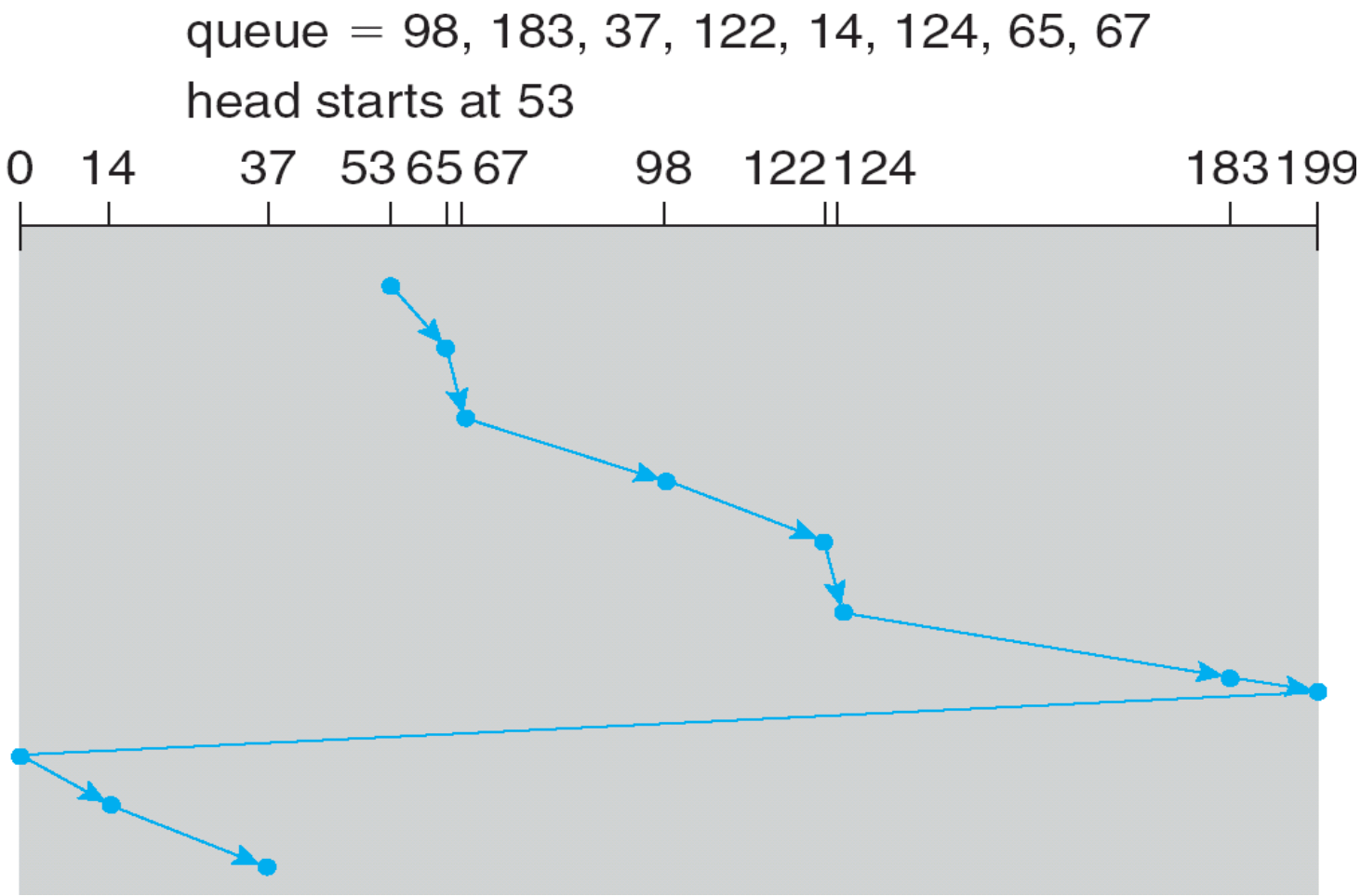
Scan

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both ends.
- Tends to stay more at the ends so more fair to the extreme cylinder requests.

Look

- The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.

C-Scan Example

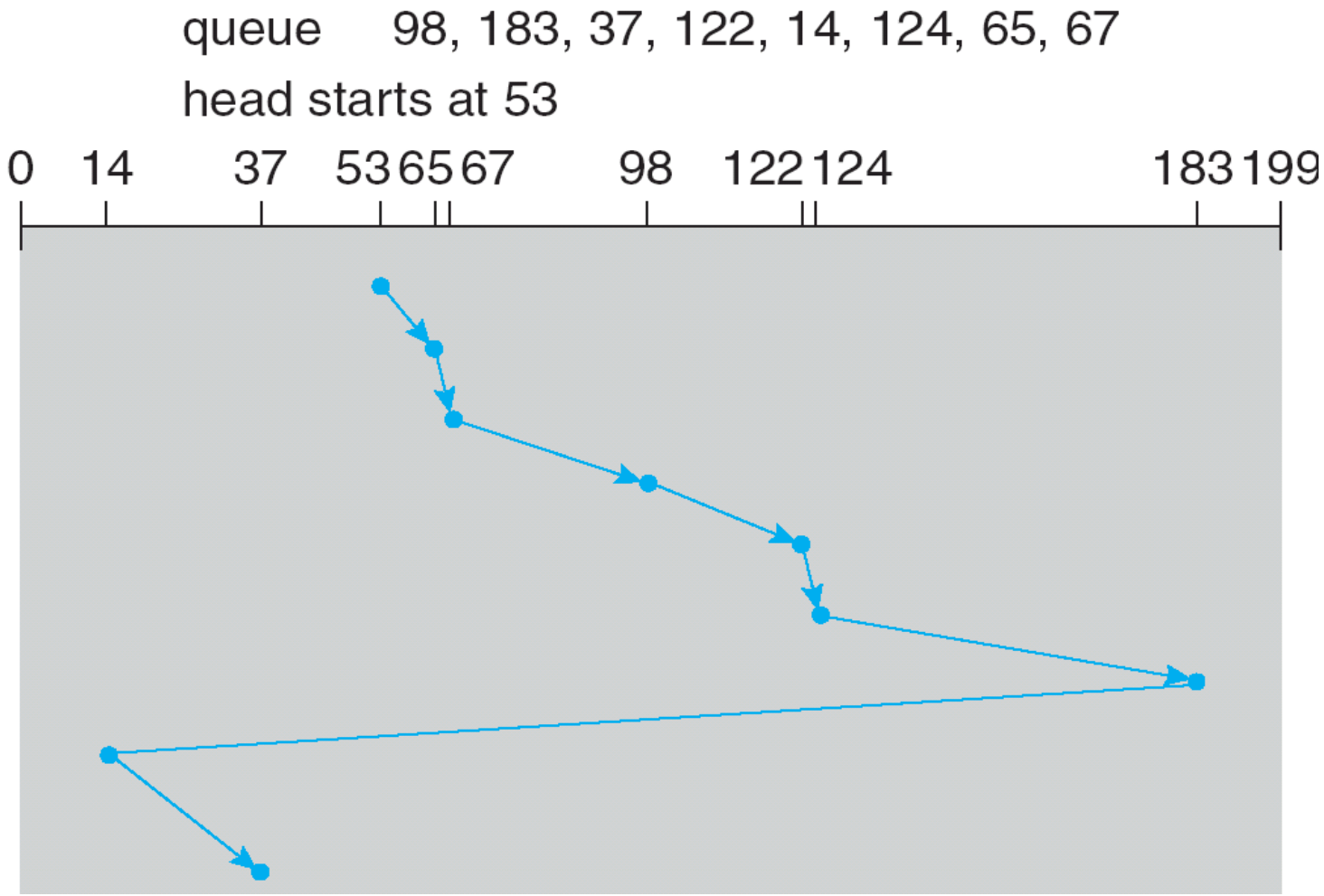


ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

C-Scan

- The head moves from one end of the disk to the other, servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
- Provides a more uniform wait time than SCAN; it treats all cylinders in the same manner.

C-Look Example



ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

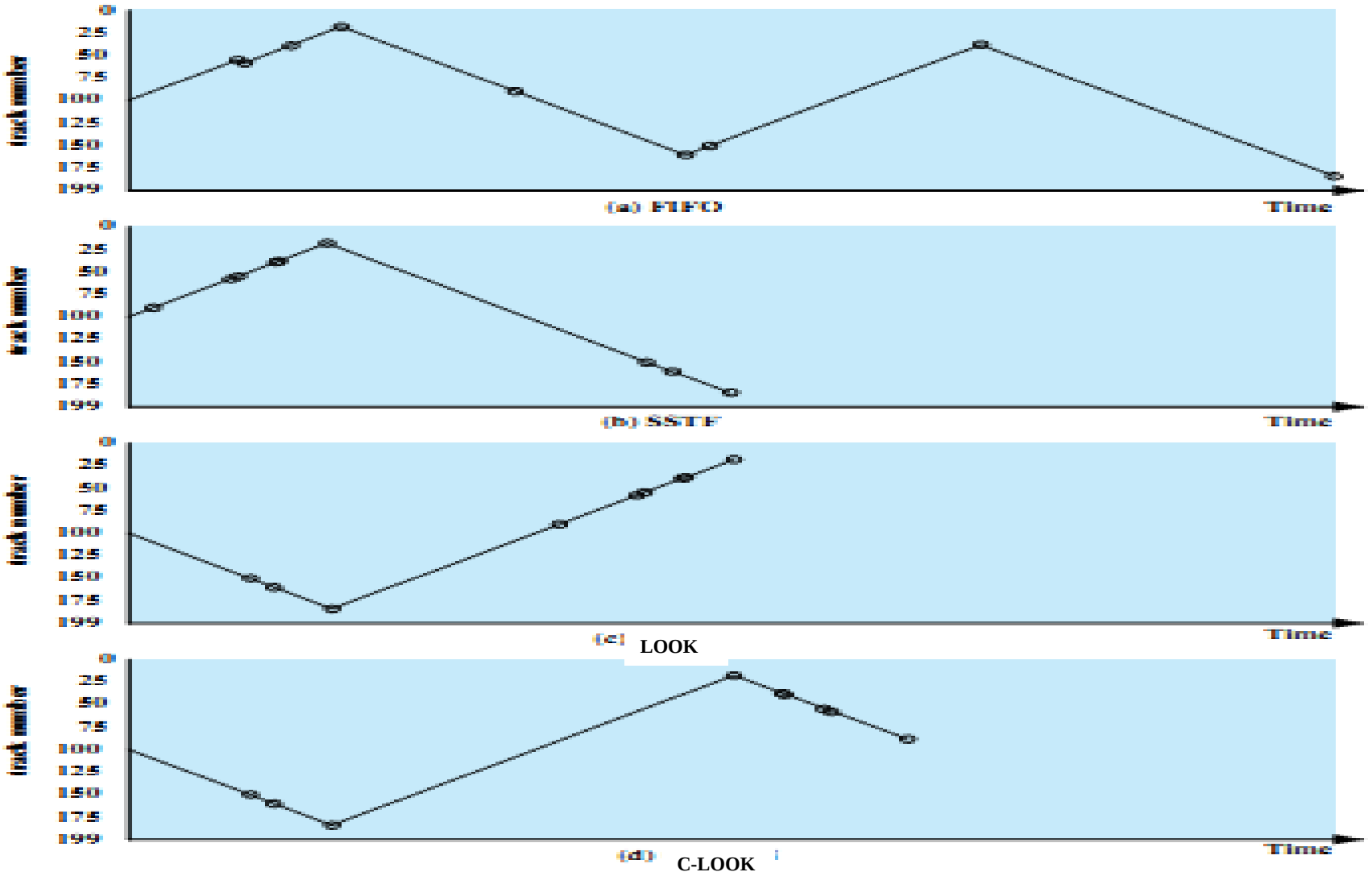
C-Look

- Look version of C-Scan.
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- In general, Circular versions are more fair but pay with a larger total seek time.
- Scan versions have a larger total seek time than the corresponding Look versions.

Another Example

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) LOOK (starting at track 100, in the direction of increasing track number)		(d) C-LOOK (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Graphs for previous example



Other Disk Scheduling Policies

- Pickup
  - A combination of FCFS and Look.
  - Goes to next I/O request by FCFS but services all existing requests on the way to it.
- Priority
  - Goal is not to optimize disk use but to meet other objectives.
  - Short batch jobs may have higher priority.
  - Provide good interactive response time.

Scan Algorithm Variations

- FScan
  - Use two queues.
  - One queue is empty to receive new requests.
- N-step-Scan
  - Segments the disk request queue into subqueues of length N.
  - Subqueues are processed one at a time, using Scan.
  - New requests added to other queue when a certain queue is processed.

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Selecting a Disk-Scheduling Algorithm (1)

- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Selecting a Disk-Scheduling Algorithm (2)

- With low load on the disk, It's FCFS anyway.
- SSTF is common and has a natural appeal – good for medium disk load.
  - SCAN and C-SCAN perform better for systems that place a heavy load on the disk; Less starvation.
  - Performance depends on number and types of requests.
- Requests for disk service can be influenced by the file-allocation method and metadata layout.
- Either SSTF or LOOK (as part of an Elevator package) is a reasonable choice for the default algorithm.

SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY