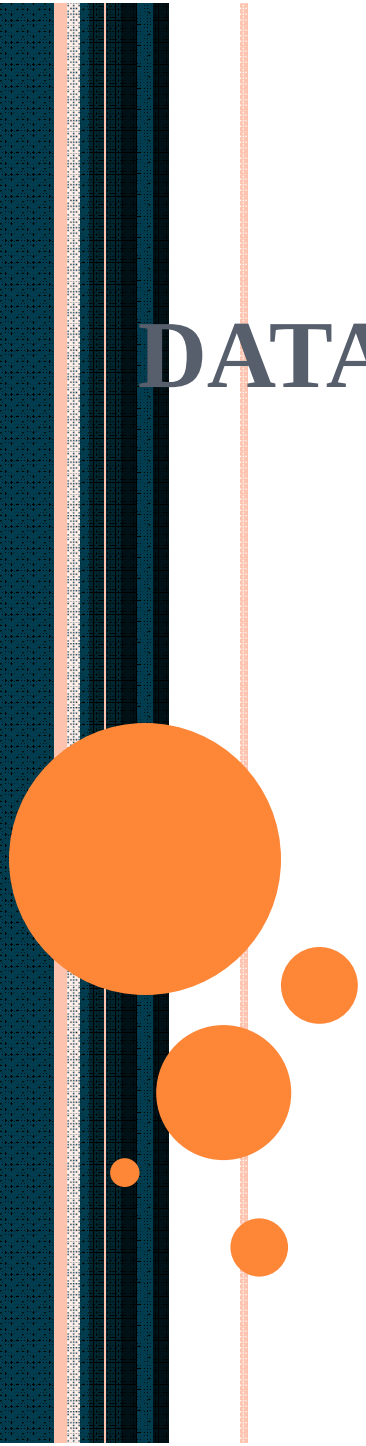
DATA STRUCTURES USING ‘C’



File Management

Chapter 9

File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

File Attributes

- **Name** – the only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file 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

File Management

- File management system consists of system utility programs that run as privileged applications
- Input to applications is by means of a file
- Output is saved in a file for long-term storage

File System Properties

- Long-term existence
  - Stored on disk or secondary/tertiary storage
- Sharable between processes
  - Access can be controlled, with permissions
- Structure
  - Depending on the file structure, a file can have internal structure convenient for a particular application.
  - Files can be organized in hierarchy or more complex structure – to reflect relationships among them.

File Operations

- Create – define new file and position it within file structure.
- Delete – remove from the file structure and destroyed.
- Open – to allow a process to perform functions on it.
- Close – close with respect to a process.
- Read – read all or a portion of a file.
- Write (update) – add new data, or change values.

Terms Used with Files

- Field
  - Basic element of data
  - Contains a single value
  - Characterized by its length and data type
- Record
  - Collection of related fields
  - Treated as a unit
    - Example: employee record (Fields: name, emp_num, job_class)
  - May be fixed or variable length

Terms Used with Files

- File
  - Collection of similar records
  - Treated as a single entity
  - Have file names
  - May restrict access
- Database
  - Collection of related data
  - Relationships exist among elements

Typical Operations

- Retrieve_All
- Retrieve_One
- Retrieve_Next
- Retrieve_Previous
- Insert_One
- Delete_One
- Update_One
- Retrieve_Few

File Management Systems

- A set of system software.
- The way a user of application may access files is through the FMS
- Programmer does not need to develop file management software

Objectives for a File Management System

- Meet the data management needs and requirements of the user
  - Storage, ability to perform operations
- Guarantee that the data in the file are valid
- Optimize performance
  - System throughput, response time (user's view)
- Provide I/O support for a variety of storage device types

Objectives for a File Management System

- Minimize or eliminate the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines to user processes
- Provide I/O support for multiple users

Minimal Set of Requirements

- Each user should be able to create, delete, read, write and modify files
- Each user may have controlled access to other users' files
- Each user may control what type of accesses are allowed to the users' files
- Each user should be able to restructure the user's files in a form appropriate to the problem

Minimal Set of Requirements

- Each user should be able to move data between files
- Each user should be able to back up and recover the user's files in case of damage
- Each user should be able to access the user's files by using symbolic names

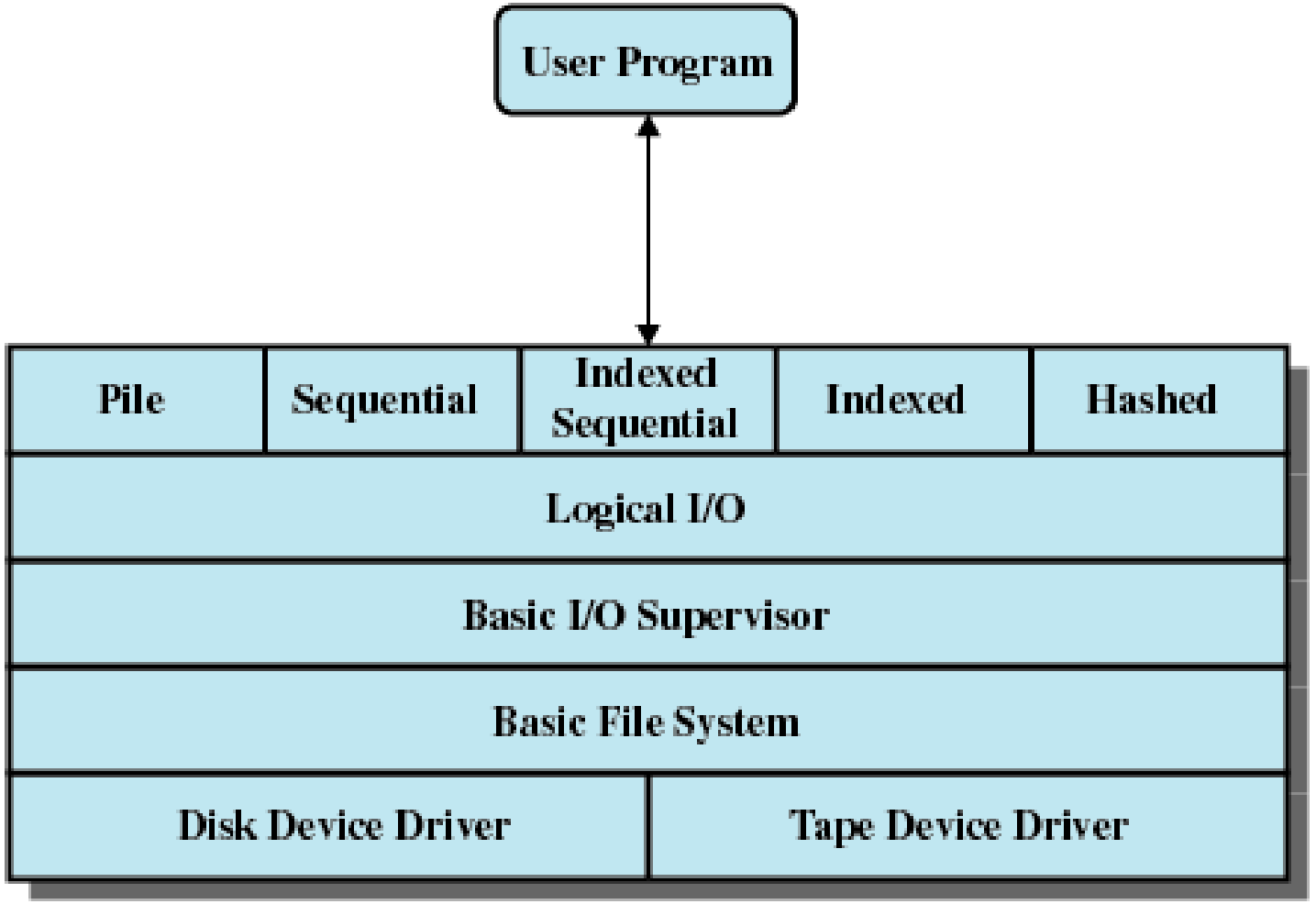


Figure 12.1 File System Software Architecture

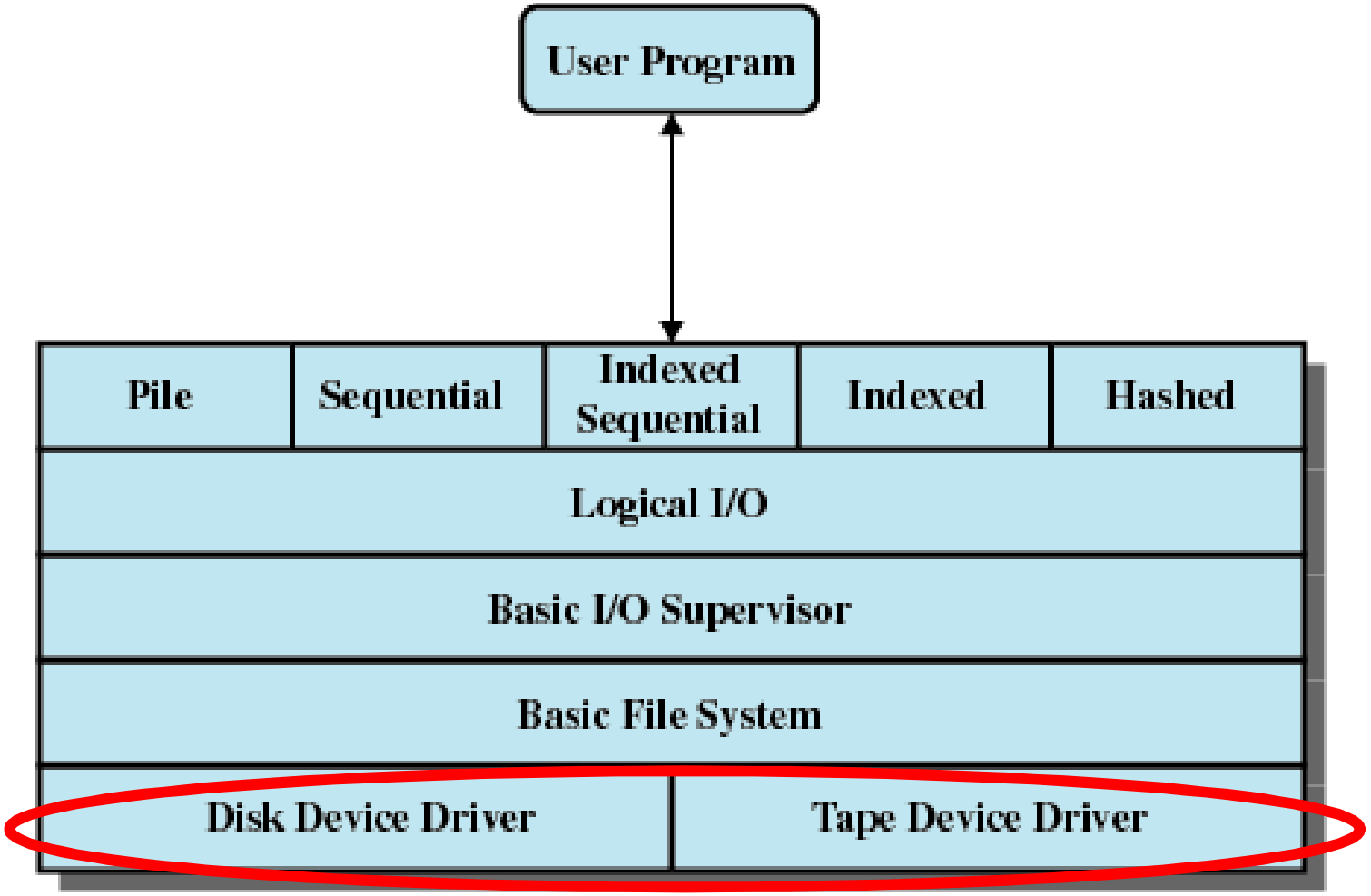


Figure 12.1 File System Software Architecture

Device Drivers

- Lowest level
- Communicates directly with peripheral devices or their controllers or channels
- Responsible for starting I/O operations on a device
- Processes the completion of an I/O request
- Typical device controlled (for file operation):
  - disk drives, tape drives
- Usually considered as part of OS

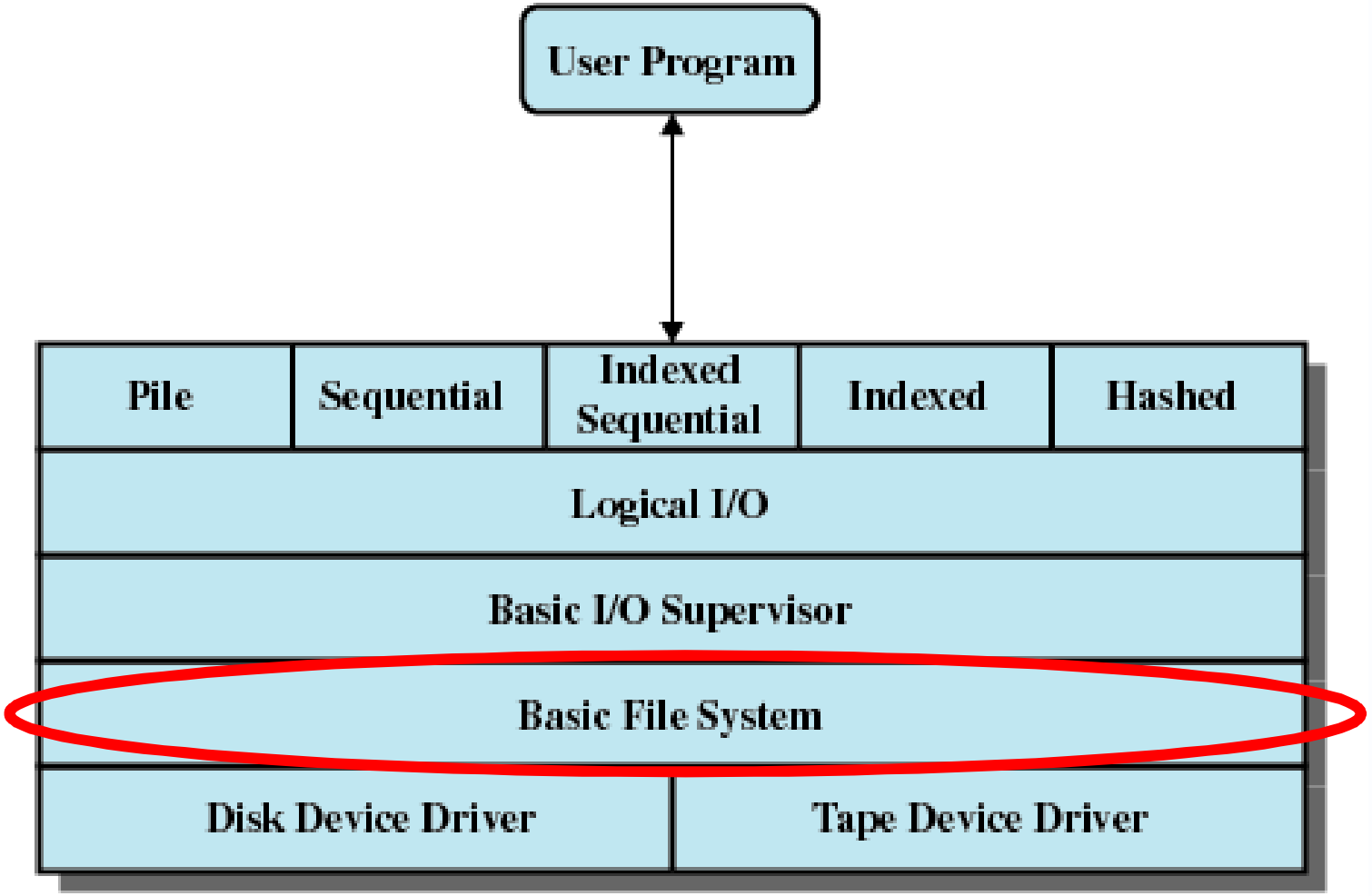


Figure 12.1 File System Software Architecture

Basic File System

- A.k.a Physical I/O
- Deals with exchanging blocks of data
- Concerned with the placement of blocks
- Concerned with buffering blocks in main memory
- Does not understand the content of data or the structure of the files involved.
- Also part of the OS.

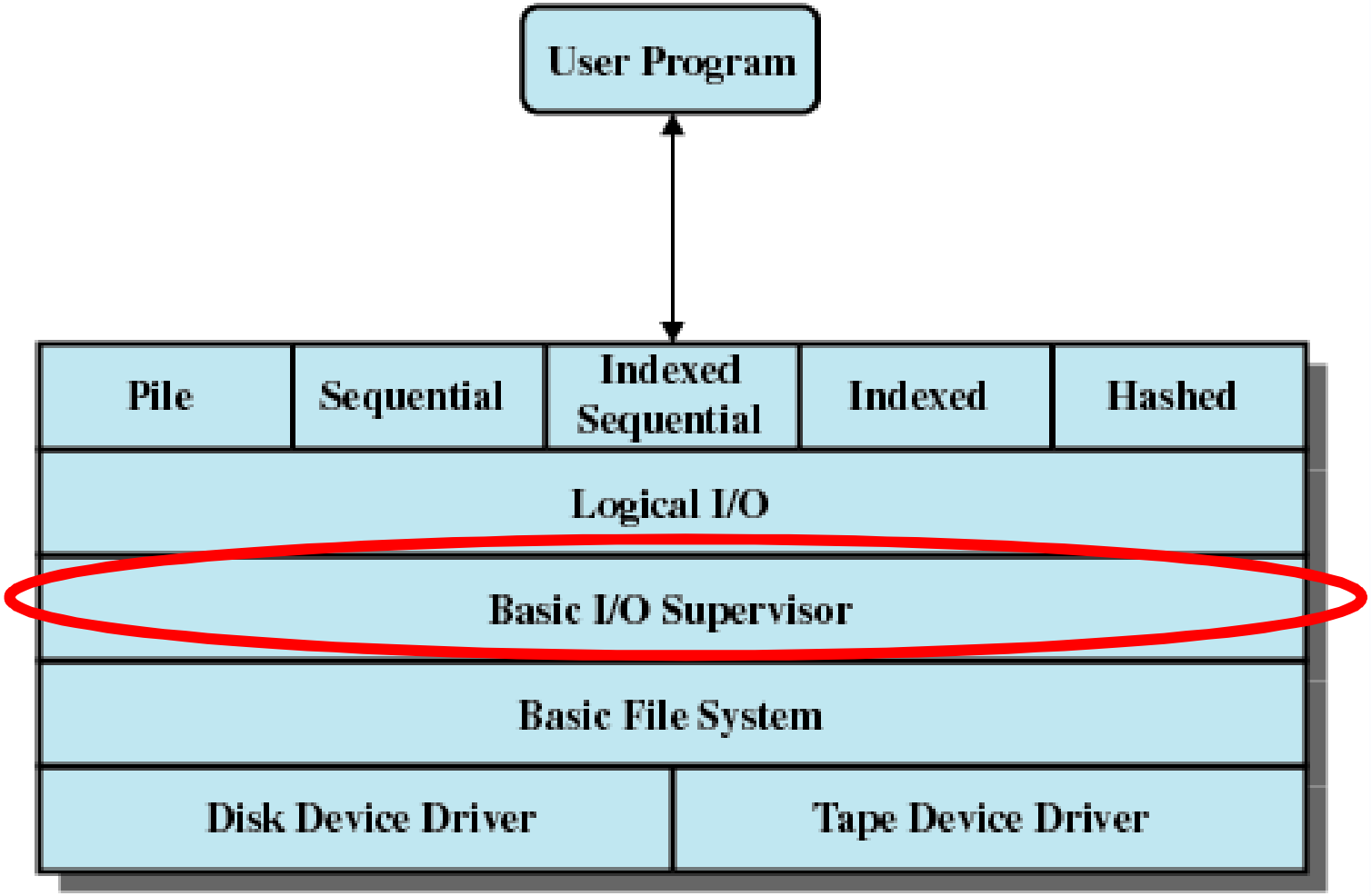


Figure 12.1 File System Software Architecture

Basic I/O Supervisor

- Responsible for file I/O initiation and termination
- Control structures are maintained
- Concerned with selection of the device on which file I/O is to be performed
- Concerned with scheduling access to optimize performance
- Part of the operating system

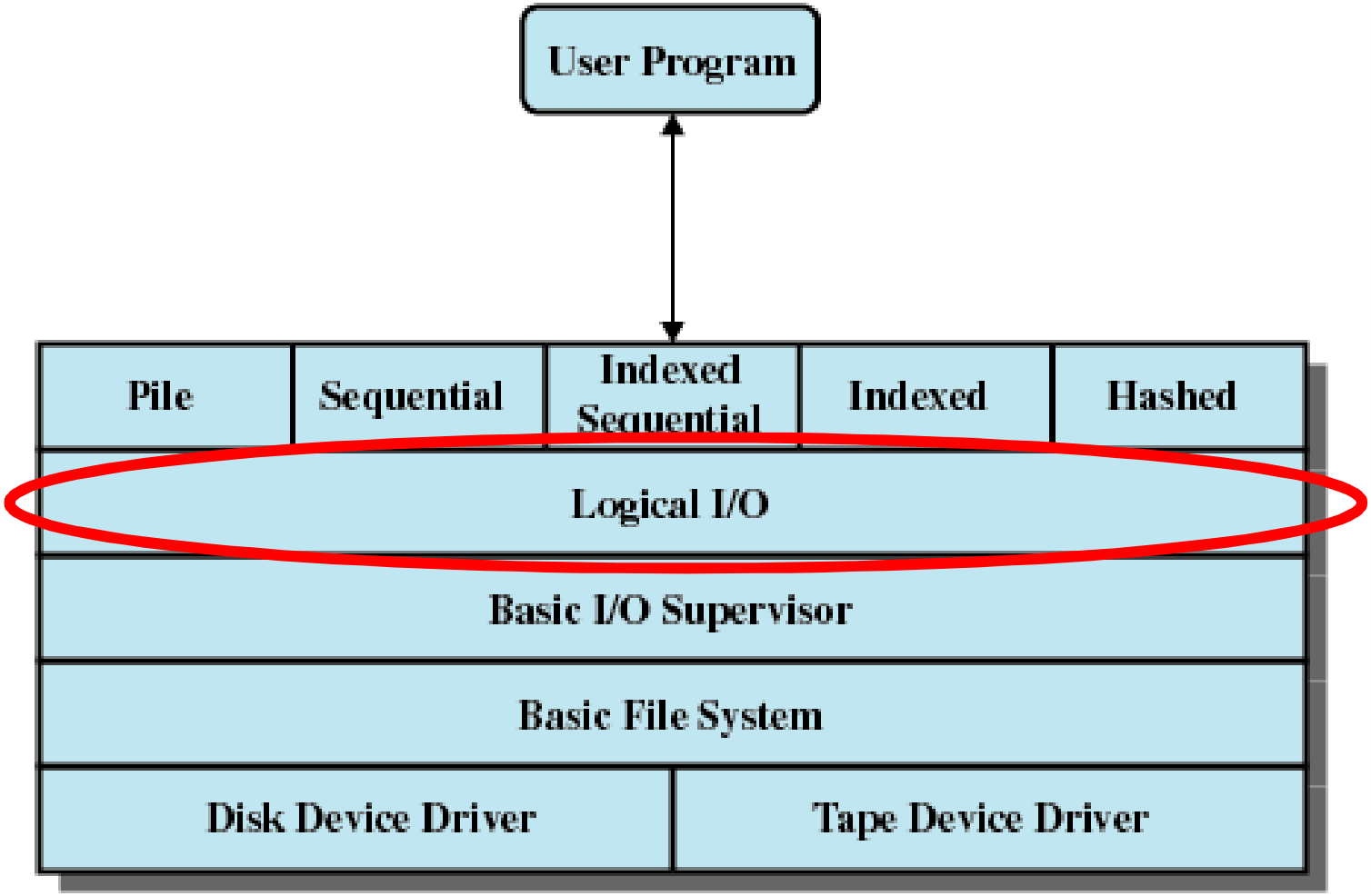


Figure 12.1 File System Software Architecture

Logical I/O

- Enables users and applications to access records
  - Thus, whereas the basic file system deals with blocks of data, the logical I/O module deals with file records.
- Provides general-purpose record I/O capability
- Maintains basic data about file

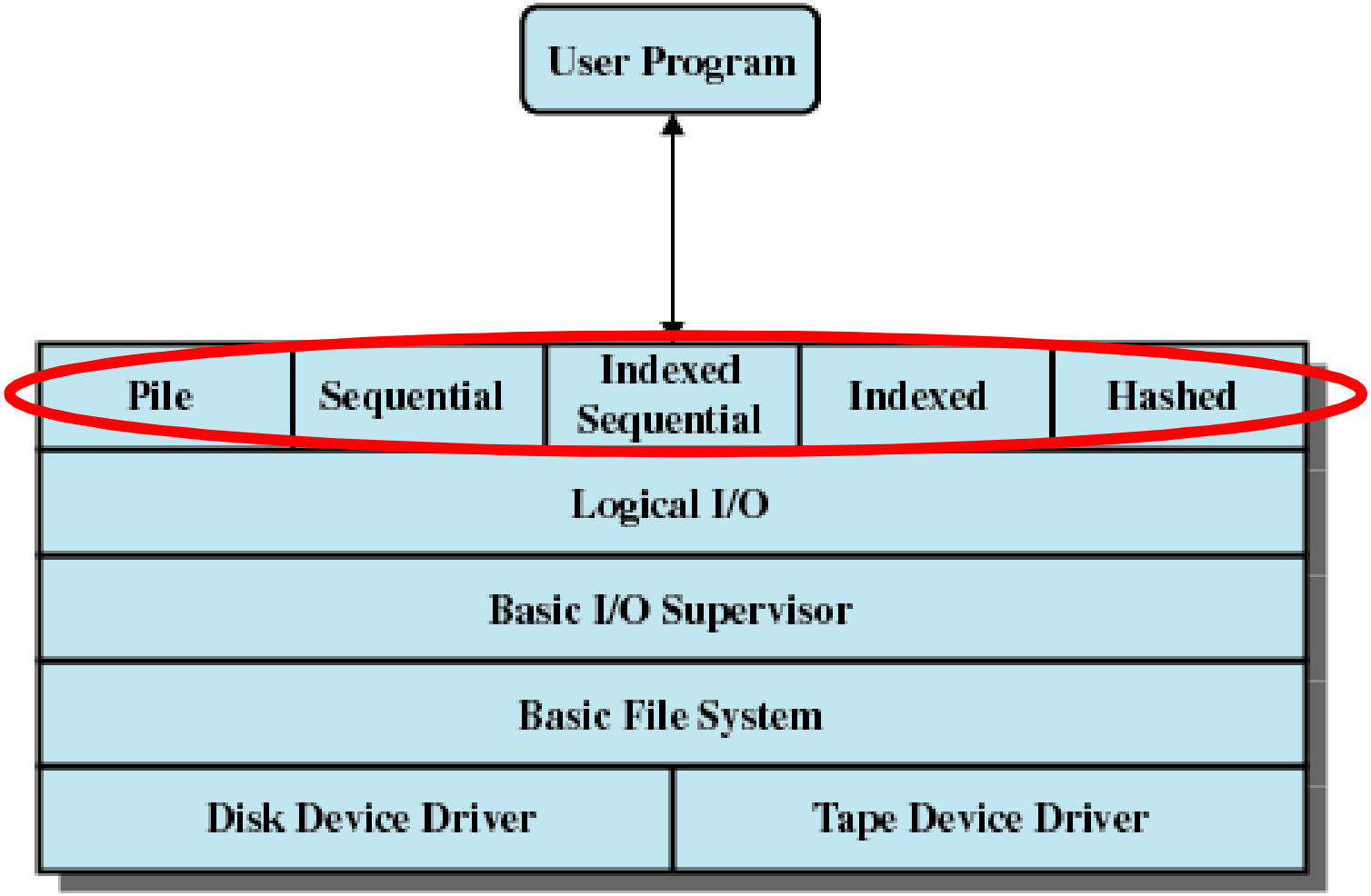


Figure 12.1 File System Software Architecture

Access Method

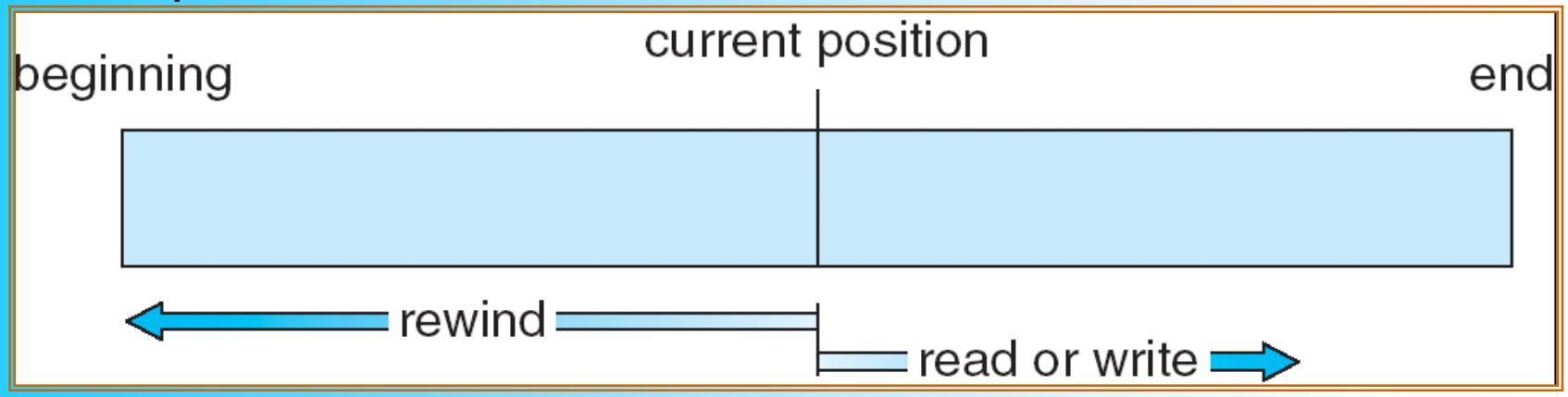
- The level of file system closest to the user is often termed as access method
- Reflect different file structures
- Different ways to access and process data
- Provides standard interface between applications and the file system and the devices that hold the data.

Access Methods

- **Sequential Access**
  - read next
  - write next
  - reset
  - no read after last write
    - (rewrite)
- **Direct Access**
  - read *n*
  - write *n*
  - position to *n*
    - read next
    - write next
  - rewrite *n*

n = relative block number

Sequential access



File Organization

- … is the logical structuring of records as how they are accessed.
- 5 structures:
  - Pile
  - Sequential file
  - Indexed sequential file
  - Indexed file
  - Direct or hashed file

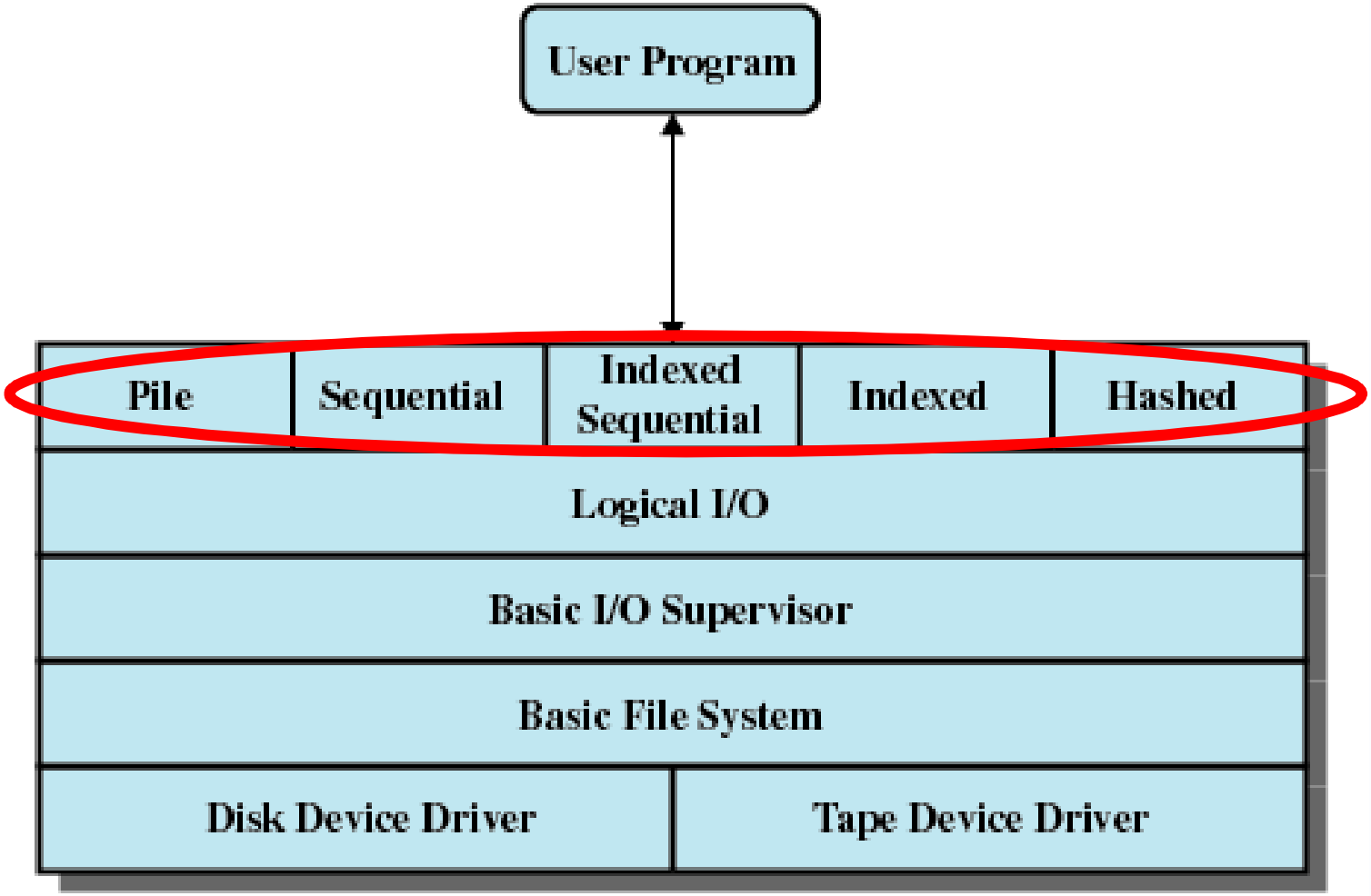
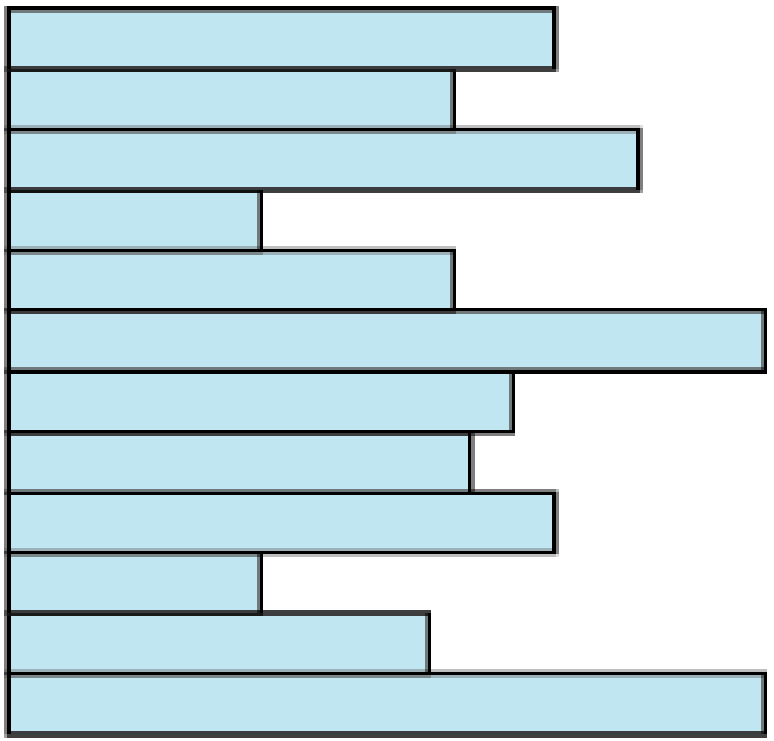


Figure 12.1 File System Software Architecture

The Pile

- Least complicated form
- Data are collected in the order they arrive
- Purpose is to accumulate a mass of data and save it
- Records may have different fields
- No structure
- Record access is by exhaustive search
- Easy to update
- But unsuitable for most applications
- Used when data are collected before processing,
- Or when data are not easy to organize
  - Uses space well

Pile



Variable-length records
Variable set of fields
Chronological order

(a) Pile File

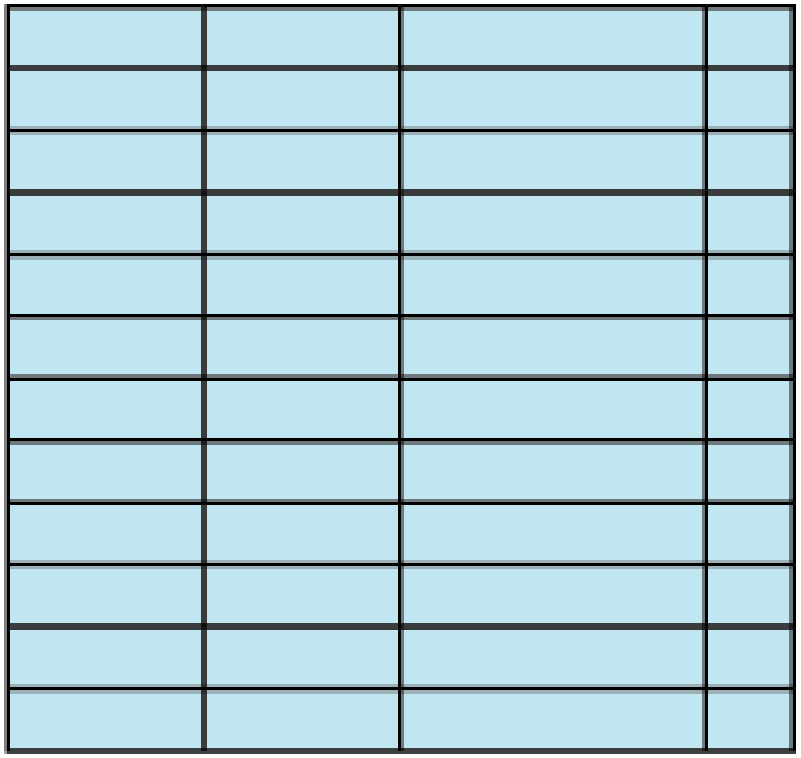
The Sequential File

- Most common
- Fixed format used for records
- Records are the same length
- All fields the same (order and length)
- Field names and lengths are attributes of the file
- One field is the key field (usually the first)
  - Uniquely identifies the record
  - Records are stored in key sequence
- New records are placed in a log file or transaction file
- Batch update is performed to merge the log file with the master file

The Sequential File

- Used in batch applications – optimum if involve processing all records
  - E.g. billing, or payroll applications
- The only file organization that can be stored on tape (as well as disk)
- Poor performance in terms of searching.

Sequential File



Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

Indexed Sequential File

- Index provides a lookup capability to quickly reach the vicinity of the desired record
  - Contains key field and a pointer to the main file
  - Indexed is searched to find highest key value that is equal to or precedes the desired key value
  - Search continues in the main file at the location indicated by the pointer

Indexed Sequential File

- A popular approach to overcome the disadvantages of sequential file.
- Maintains the key characteristics of sequential file – records are organized in sequence based on the key field.

File Organization

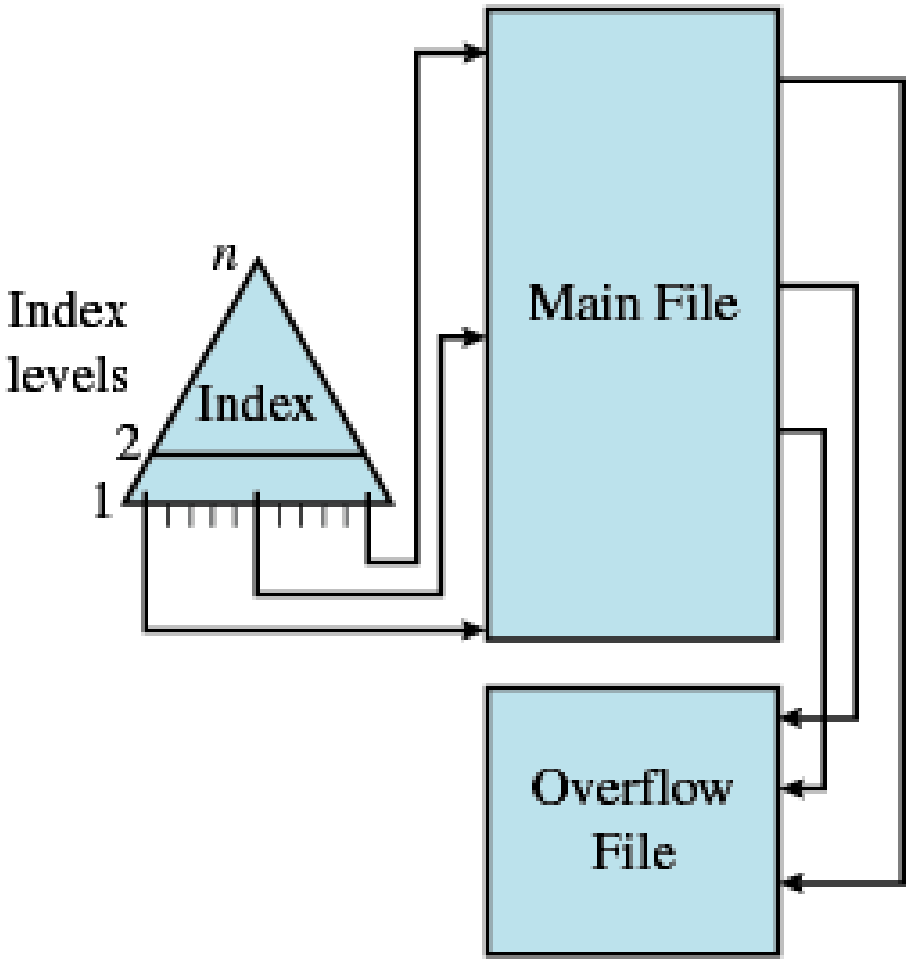
Comparison of sequential and indexed sequential

- Example: a file contains 1 million records
- **Sequential:**
  - On average 500,000 accesses are required to find a record in a sequential file
- **Indexed sequential**:
  - If an index contains 1000 entries, it will take on average 500 accesses to find the key, followed by 500 accesses in the main file. Now on average it is 1000 accesses.

Indexed Sequential File

- New records are added to an overflow file
- Record in main file that precedes it is updated to contain a pointer to the new record
- The overflow is merged with the main file during a batch update
- Multiple indexes for the same key field can be set up to increase efficiency

Indexed Sequential File

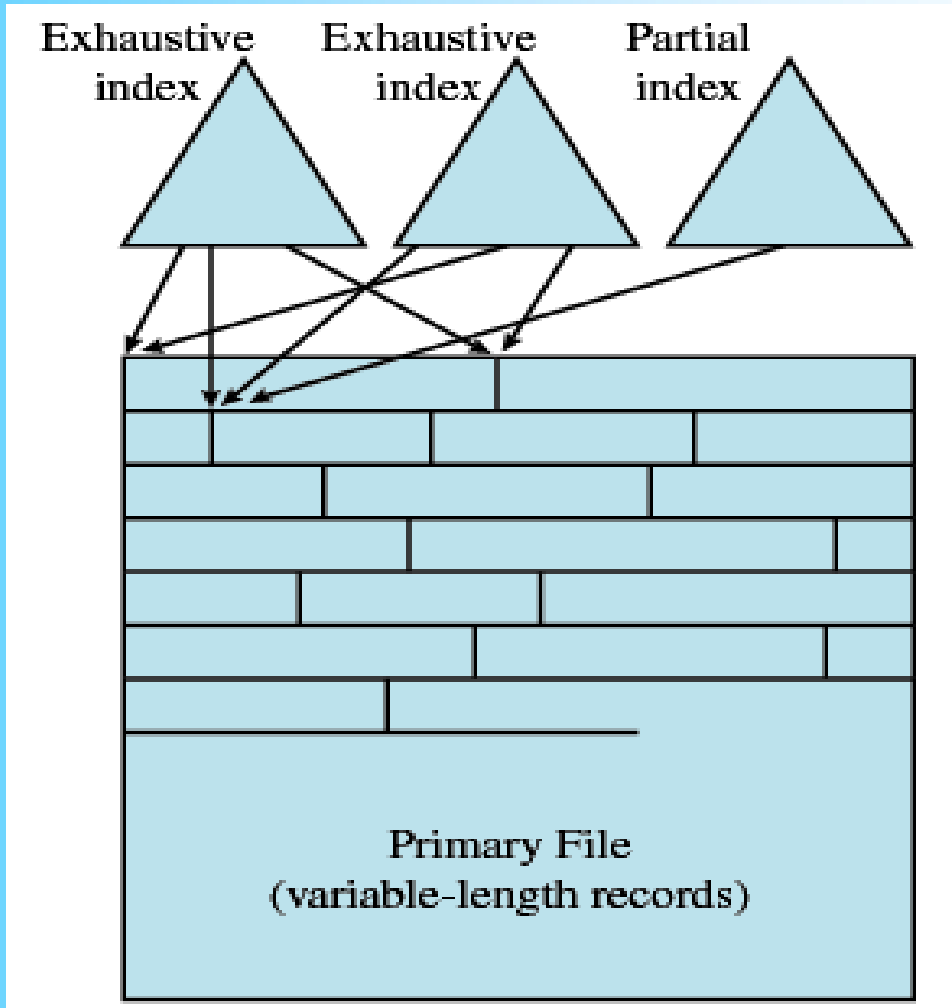


(c) Indexed Sequential File

Indexed File

- Uses multiple indexes for different key fields
- May contain an exhaustive index that contains one entry for every record in the main file
  - The index is organized as a sequential file for ease of searching
- May contain a partial index – contains entries to records where the field of interest exists.
- Used where timeliness of the info is critical and where data are rarely processed exhaustively
  - E.g. airline reservation syst, inventory control syst.

Indexed File



(d) Indexed File

Direct or Hashed File

- Directly access a block at a known address
- Key field required for each record
- Used where very rapid access is required,
- Or where fixed-length records are used,
- Or where records are always accessed one at a time.
- E.g. directories, pricing tables, schedules, name lists.

Table 12.1 Grades of Performance for Five Basic File Organizations [WIED87]

	Space		Update		Retrieval		
	Attributes		Record Size				
File Method	Variable	Fixed	Equal	Greater	Single record	Subset	Exhaustive
Pile	A	B	A	E	E	D	B
Sequential	F	A	D	F	F	D	A
Indexed sequential	F	B	B	D	B	D	B
Indexed	B	C	C	C	A	B	D
Hashed	F	B	B	F	B	F	E

A	=	Excellent, well suited to this purpose	$\approx O(r)$
B	=	Good	$\approx O(o \times r)$
C	=	Adequate	$\approx O(r \log n)$
D	=	Requires some extra effort	$\approx O(n)$
E	=	Possible with extreme effort	$\approx O(r \times n)$
F	=	Not reasonable for this purpose	$\approx O(n^{>1})$

where

r = size of the result
o = number of records that overflow
n = number of records in file

File Management

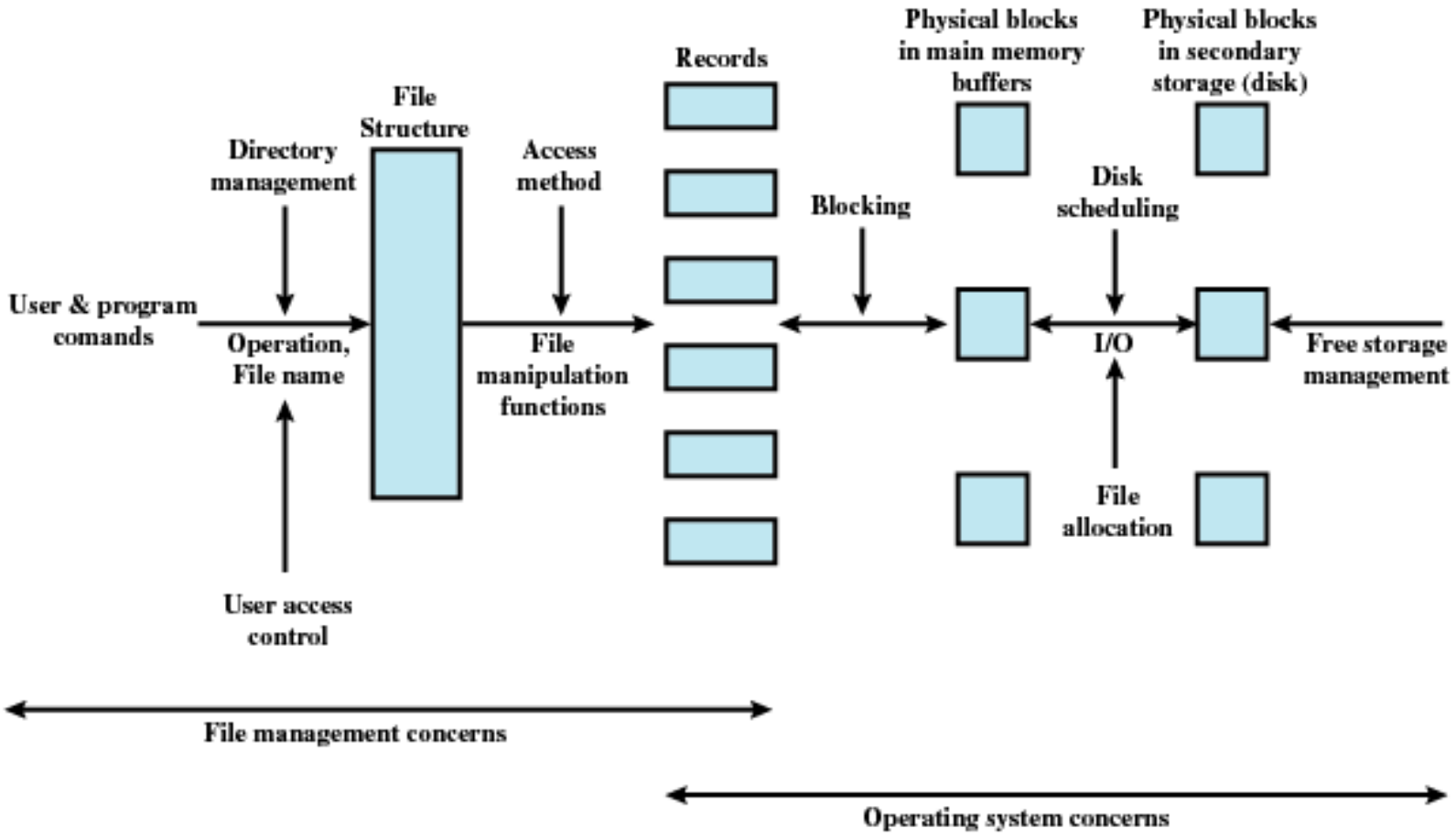


Figure 12.2 Elements of File Management

File Management Functions

- Identify and locate a selected file
- Use a directory to describe the location of all files plus their attributes
- On a shared system describe user access control
- Blocking for access to files
- Allocate files to free blocks
- Manage free storage for available blocks

Criteria for File Organization

- Short access time
  - Needed when accessing a single record
  - Not needed for batch mode
- Ease of update
  - File on CD-ROM will not be updated, so this is not a concern

Criteria for File Organization

- Economy of storage
  - Should be minimum redundancy in the data
  - Redundancy can be used to speed access such as an index
- Simple maintenance
- Reliability

File Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- Directory itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

Table 12.2 Information Elements of a File Directory

	Basic Information
File Name	Name as chosen by creator (user or program). Must be unique within a specific directory.
File Type	For example: text, binary, load module, etc.
File Organization	For systems that support different organizations
	Address Information
Volume	Indicates device on which file is stored
Starting Address	Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
Size Used	Current size of the file in bytes, words, or blocks
Size Allocated	The maximum size of the file
	Access Control Information
Owner	User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges
Access Information	A simple version of this element would include the user's name and password for each authorized user.
Permitted Actions	Controls reading, writing, executing, transmitting over a network

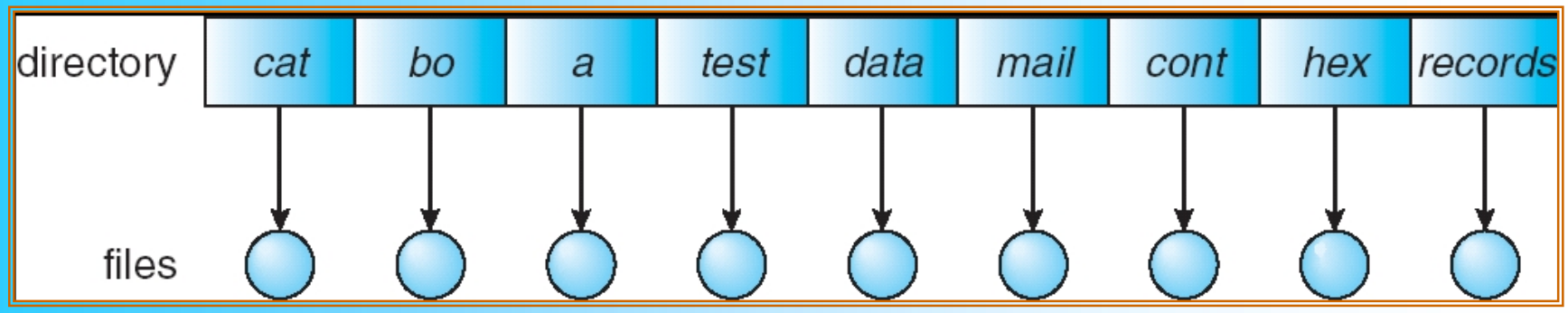
Usage Information	
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
Current Usage	Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, and whether the file has been updated in main memory but not yet on disk

Simple Structure for a Directory

- List of entries, one for each file
- Sequential file with the name of the file serving as the key
- Provides no help in organizing the files
- Forces user to be careful not to use the same name for two different files

Single-Level Directory

- A single directory for all users



Naming problem

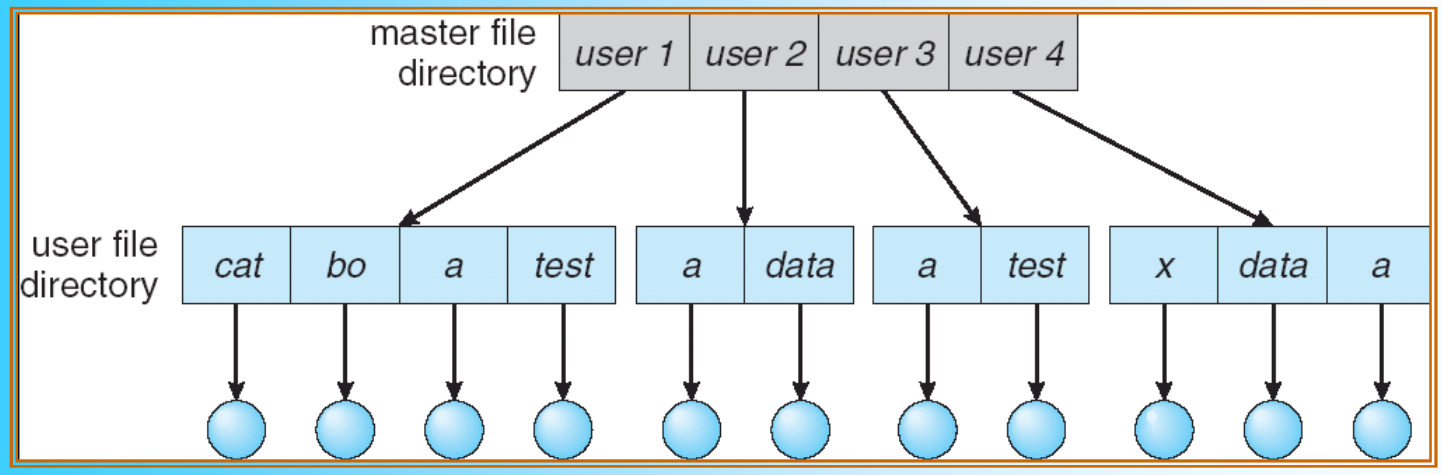
Grouping problem

Two-level Scheme for a Directory

- One directory for each user and a master directory
- Master directory contains entry for each user
  - Provides address and access control information
- Each user directory is a simple list of files for that user
- Still provides no help in structuring collections of files

Two-Level Directory

- Separate directory for each user



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

Hierarchical, or Tree-Structured Directory

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries

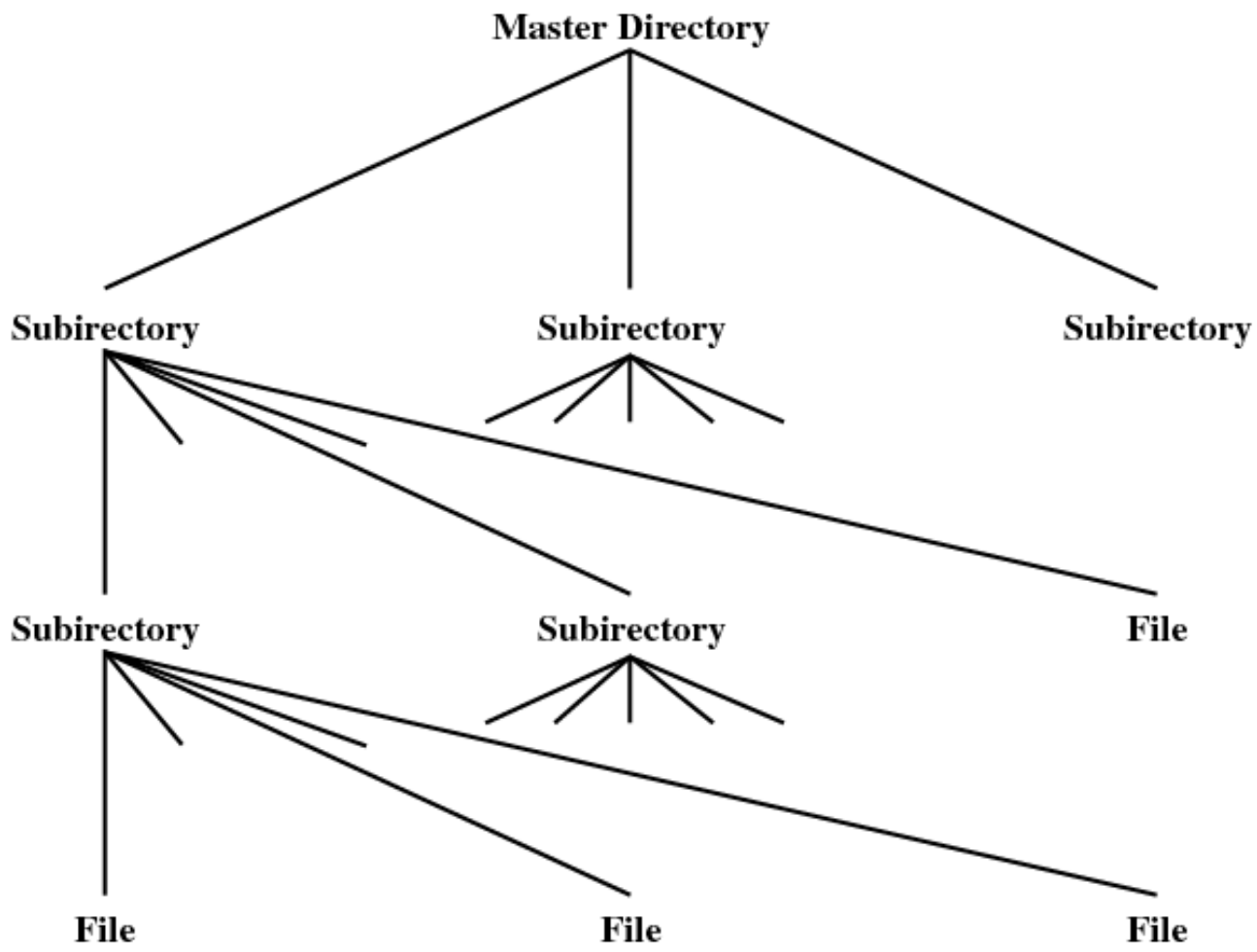
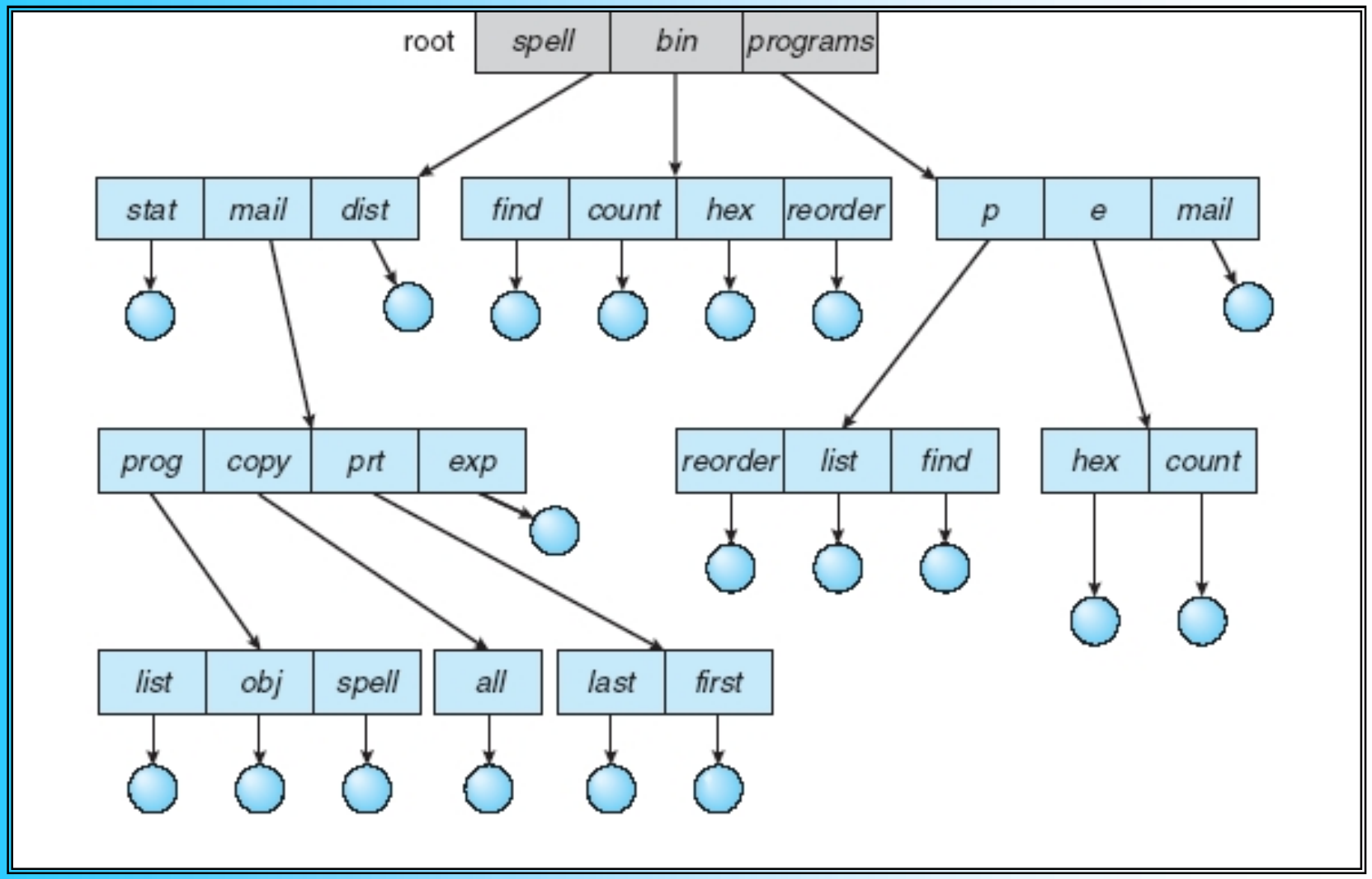


Figure 12.4 Tree-Structured Directory

Tree-Structured Directories



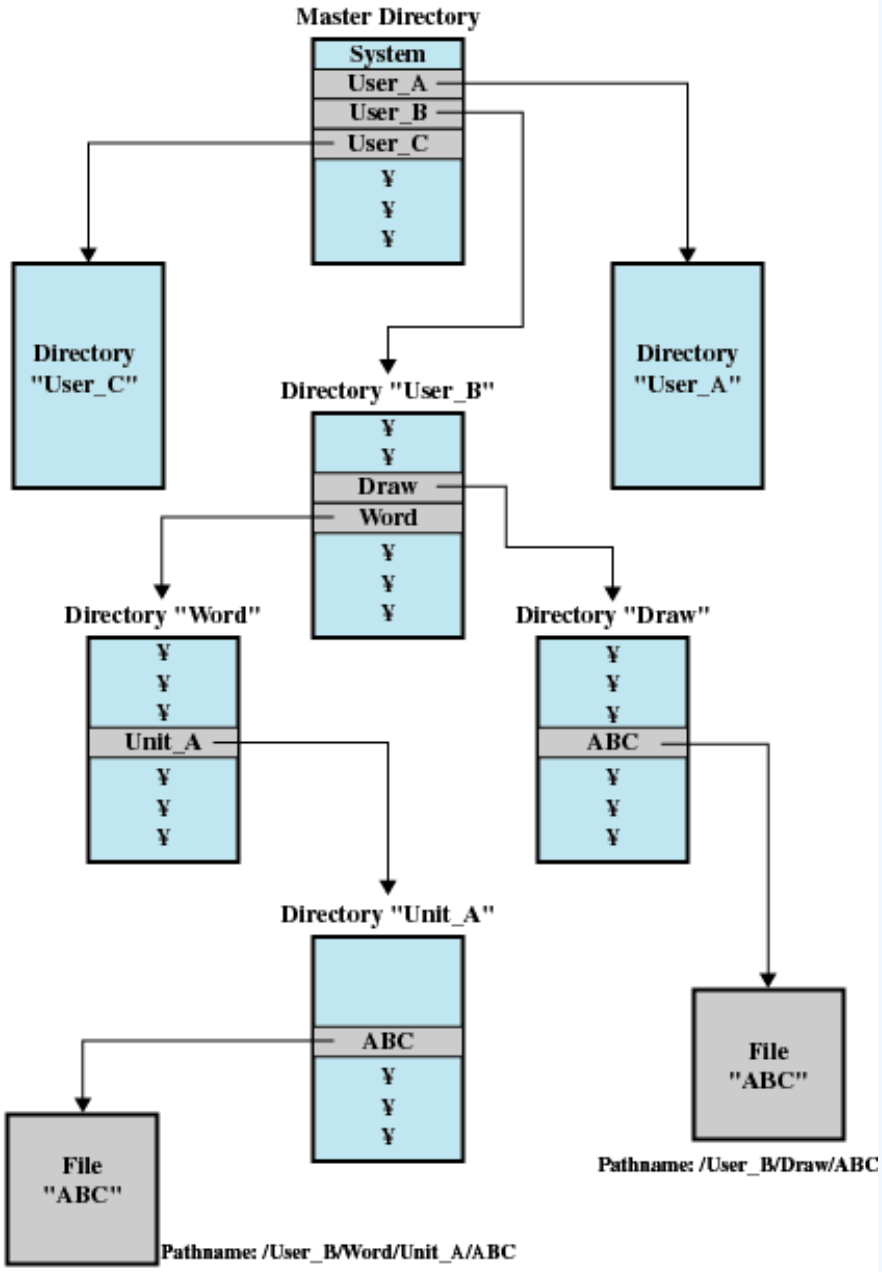


Figure 12.5 Example of Tree-Structured Directory

Hierarchical, or Tree-Structured Directory

- Files can be located by following a path from the root, or master, directory down various branches
  - This is the pathname for the file
- Can have several files with the same file name as long as they have unique path names

Hierarchical, or Tree-Structured Directory

- Current directory is the working directory
- Files are referenced relative to the working directory

Tree-Structured Directories

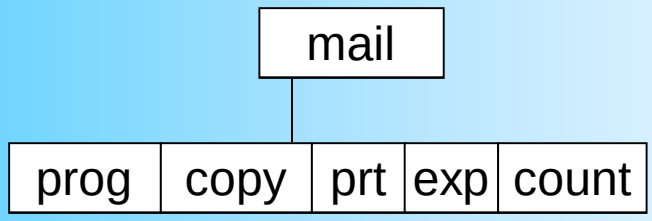
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

  rm <file-name>
- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

  mkdir count

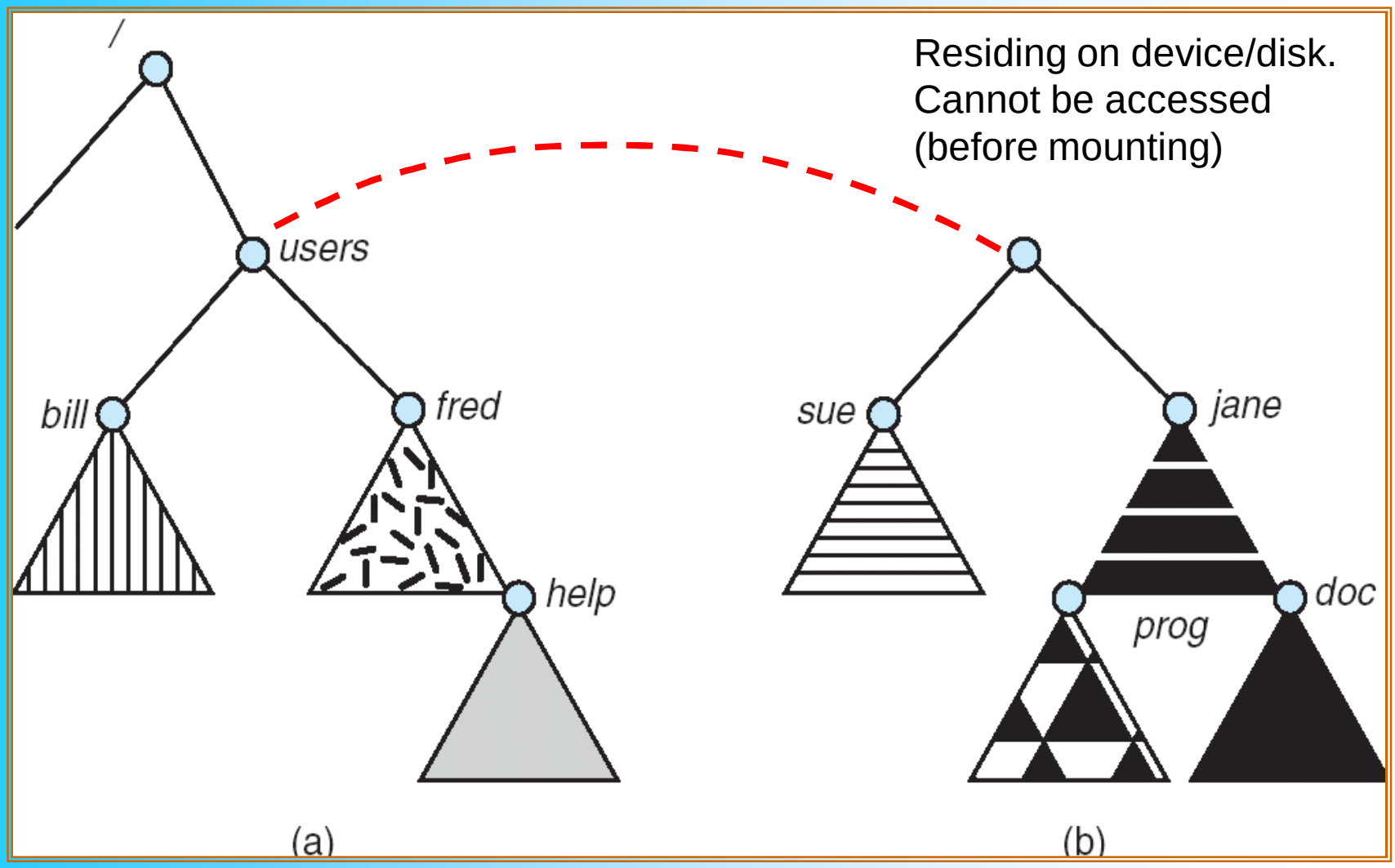


Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

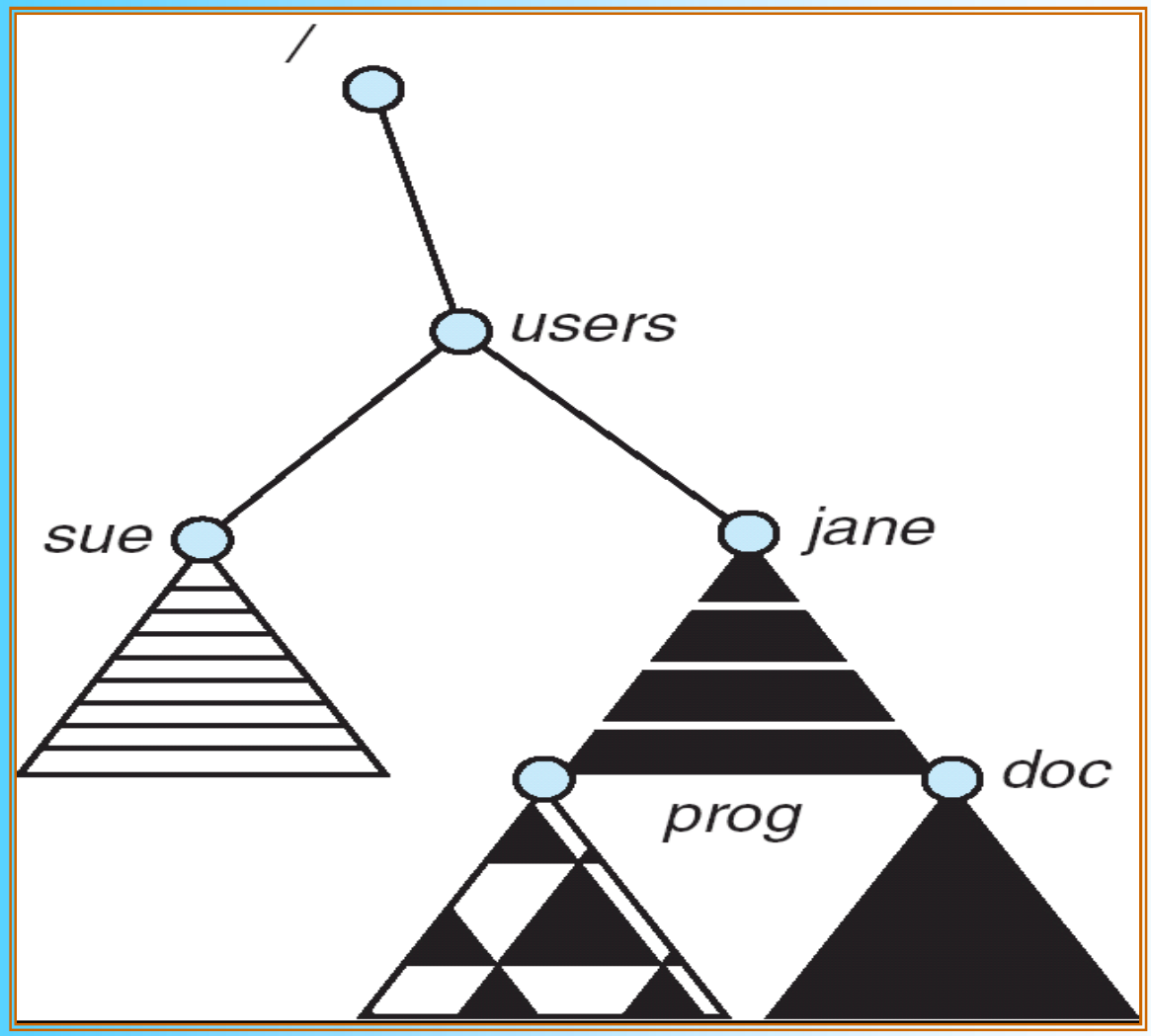
File System Mounting

- Just as a file must be opened before it can be used, a file system must be **mounted** before it can be accessed
- A unmounted file system (i.e. Fig. 11-11(b)) is mounted at a **mount point**.
- Mounting - the OS is given the name of the device and the mount point.
- The mount point is an empty directory.

(a) Existing. (b) Unmounted Partition



Mount Point



File Sharing

- In multiuser system, allow files to be shared among users
- 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

File Sharing – Multiple Users

- **User IDs** identify users, allowing permissions and protections to be per-user
- **Group IDs** allow users to be in groups, permitting group access rights

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 **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or 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

File Sharing – Failure Modes

- 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 include all information in each request, allowing easy recovery but less security

Protection

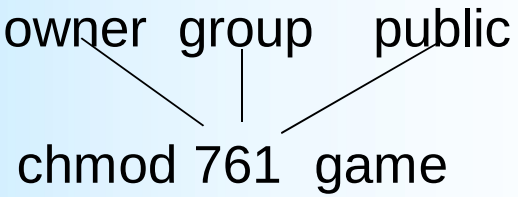
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

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

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	RWX 1 1 0
c) **public access**	1	⇒	RWX 0 0 1

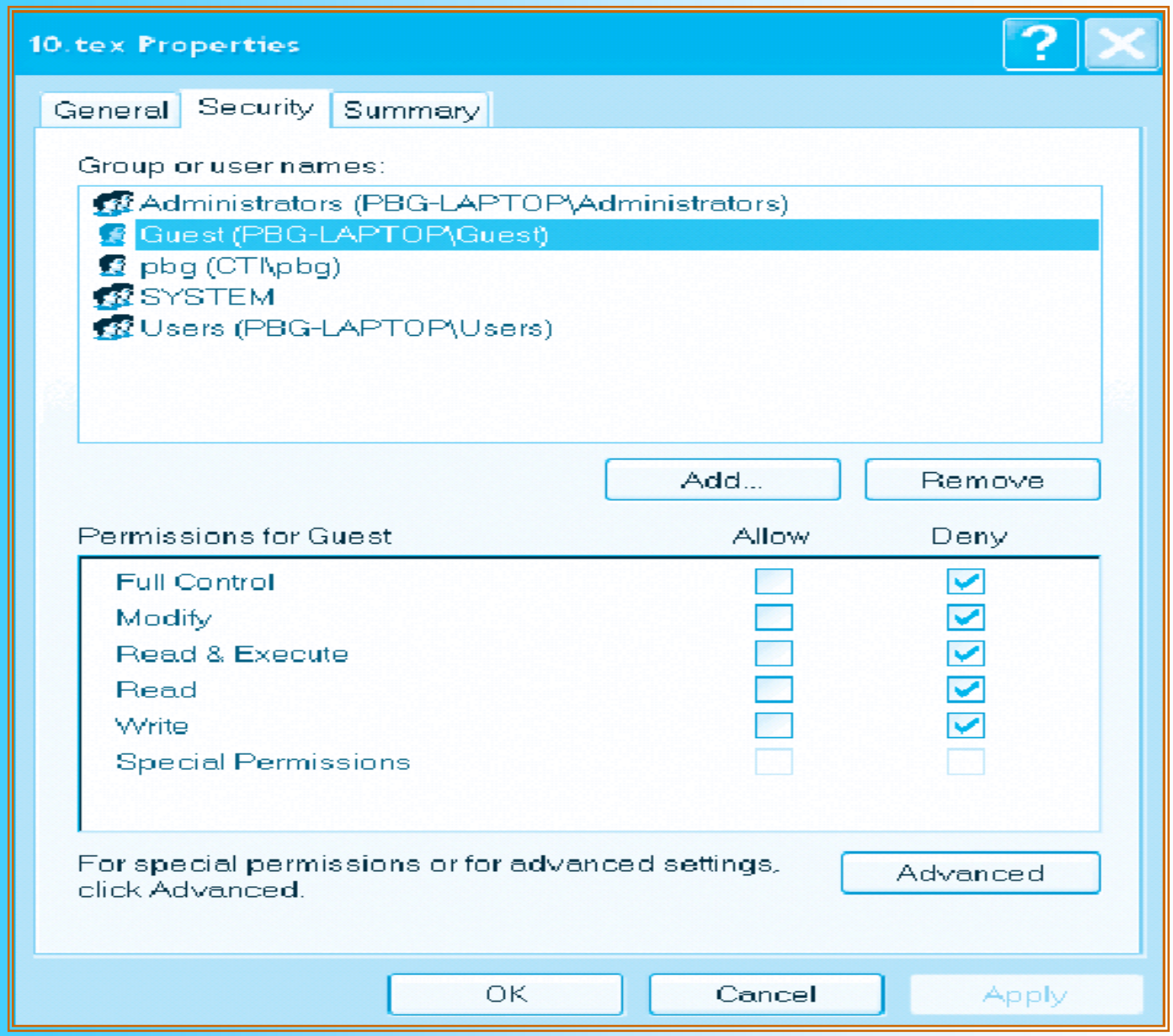
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file:

chgrp G game

Windows XP Access-control List Management



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/

File Sharing

Two issues in file sharing

- Access rights
- Management of simultaneous access

Access Rights

- None
  - User may not know of the existence of the file, much less access it
  - To enforce: User is not allowed to read the user directory that includes the file
- Knowledge
  - User can only determine that the file exists and who its owner is
  - User can then petition the owner for additional access rights

Access Rights

- Execution
  - The user can load and execute a program but cannot copy it
  - E.g. propriety program
- Reading
  - The user can read the file for any purpose, including copying and execution
  - Some system allow viewing, but not copying
- Appending
  - The user can add data to the file but cannot modify or delete any of the file's contents

Access Rights

- Updating
  - The user can modify, delete, and add to the file's data. This includes creating the file, rewriting it, and removing all or part of the data
- Changing protection
  - User can change access rights granted to other users
- Deletion
  - User can delete the file

Access Rights

- Owners
  - Has all rights previously listed
  - May grant rights to others using the following classes of users
    - Specific user
    - User groups
    - All for public files

Simultaneous Access

- User may lock entire file when it is to be updated
- User may lock the individual records during the update – finer grain
- Mutual exclusion and deadlock are issues for shared access

Record Blocking

- For I/O to be performed, records must be organized as blocks.
- Issues:
  - Should blocks be fixed or variable length?
    - Fixed on most systems
  - What should the relative size of blocks?
    - Large blocks – more records passed in one I/O operation
      - Good for sequential processing
      - Bad for random access – unnecessary transfer of unused records.
      - Also require larger buffer – difficult to manage.

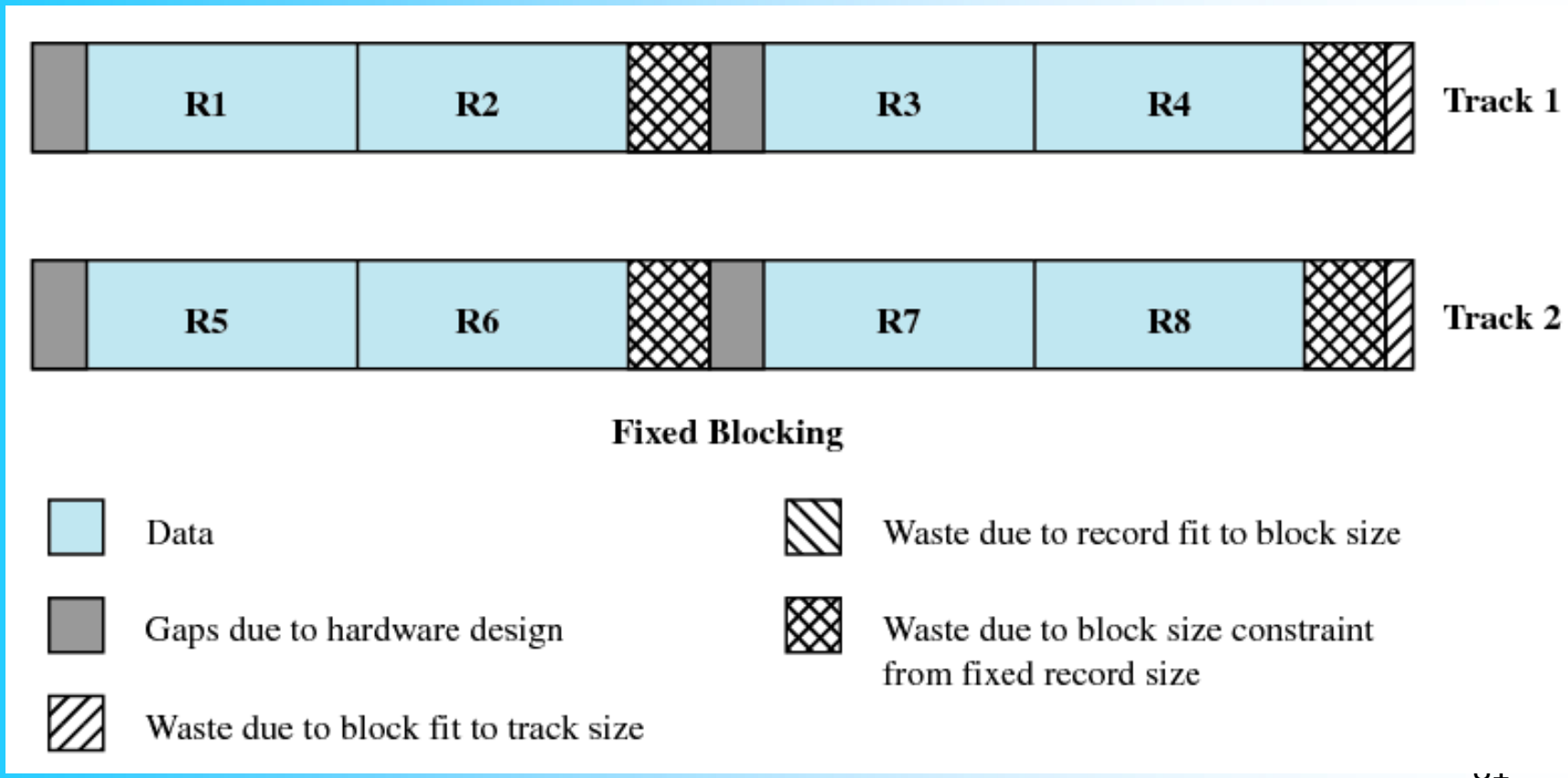
Record Blocking

Three methods of blocking:

- Fixed blocking
- Variable-length spanned blocking
- Variable-length unspanned blocking

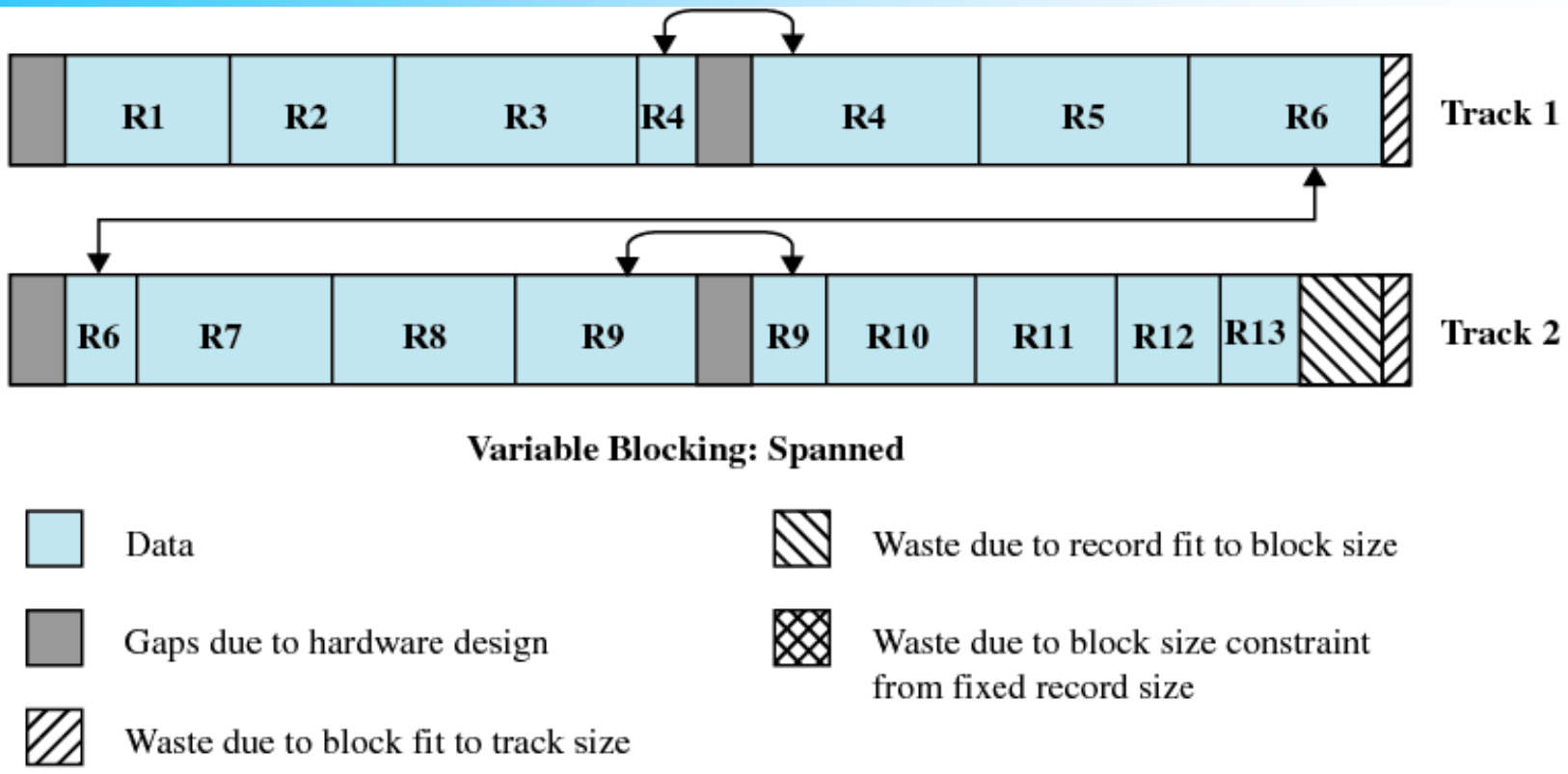
Fixed Blocking

- Fixed length records.
- An integral number of records are stored in a block
- Possible internal fragmentation.



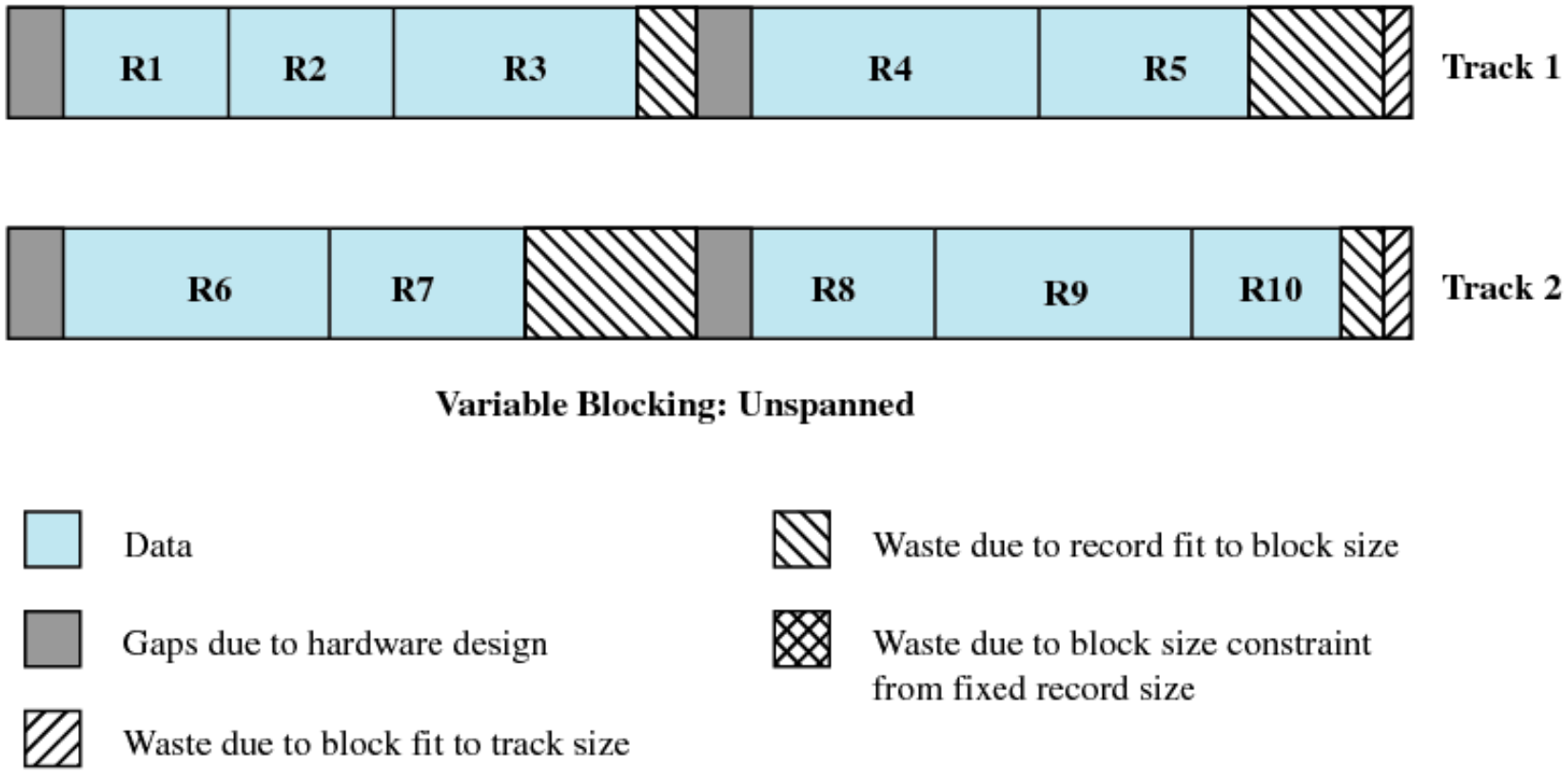
Variable Blocking: Spanned

- Variable length records, no limit to record size.
- Packed into blocks with no unused space – some records must span two blocks, with the continuation indicated by a pointer.
- (-) Records spanning 2 blocks require 2 I/O operations.



Variable Blocking Unspanned

- Variable length records.
- No spanning employed – limits record size <= block size.
- Possible internal fragmentation.



Secondary Storage Management

- Space must be allocated to files
- Must keep track of the space available for allocation
- On secondary storage, file consists of a collection of blocks.

Preallocation

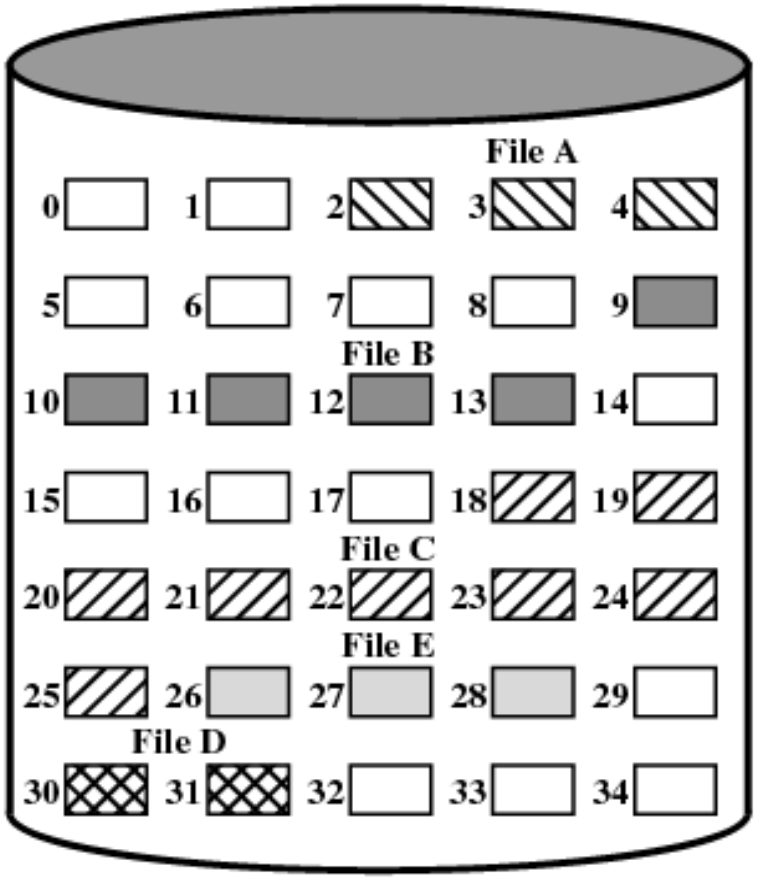
- Need the maximum size for the file at the time of creation
- Difficult to reliably estimate the maximum potential size of the file
- Tend to overestimated file size so as not to run out of space ➔ waste of unused space.
- Better to use dynamic allocation.

Methods of File Allocation

- Contiguous allocation
- Linked allocation (Chained)
- Indexed allocation

Contiguous Allocation

- Single set of blocks is allocated to a file at the time of creation
- Only a single entry in the file allocation table
  - Starting block and length of the file
- External fragmentation will occur
  - Need to perform compaction

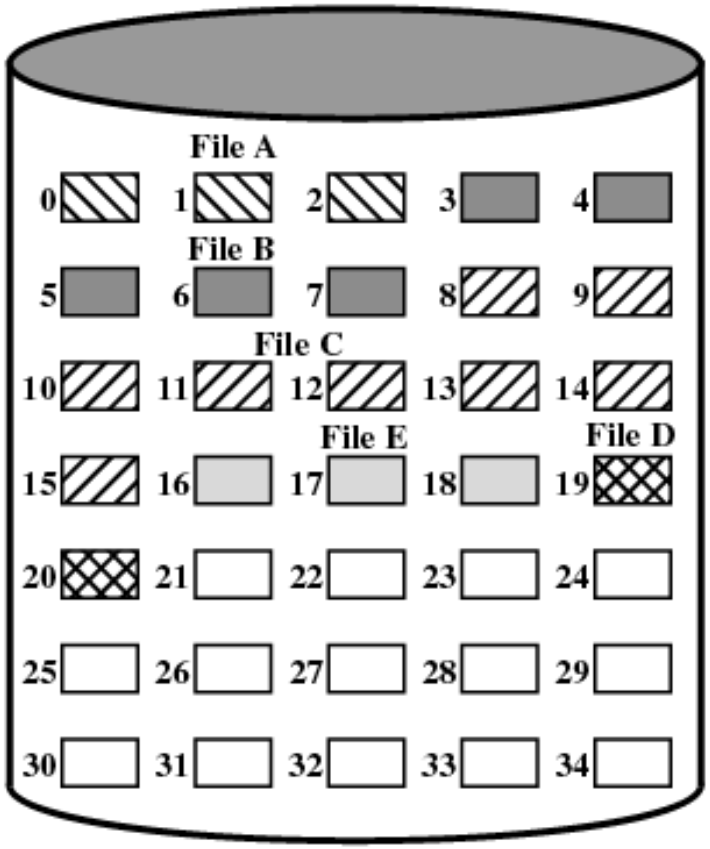


File Allocation Table

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

Leads to external fragmentation

Figure 12.7 Contiguous File Allocation

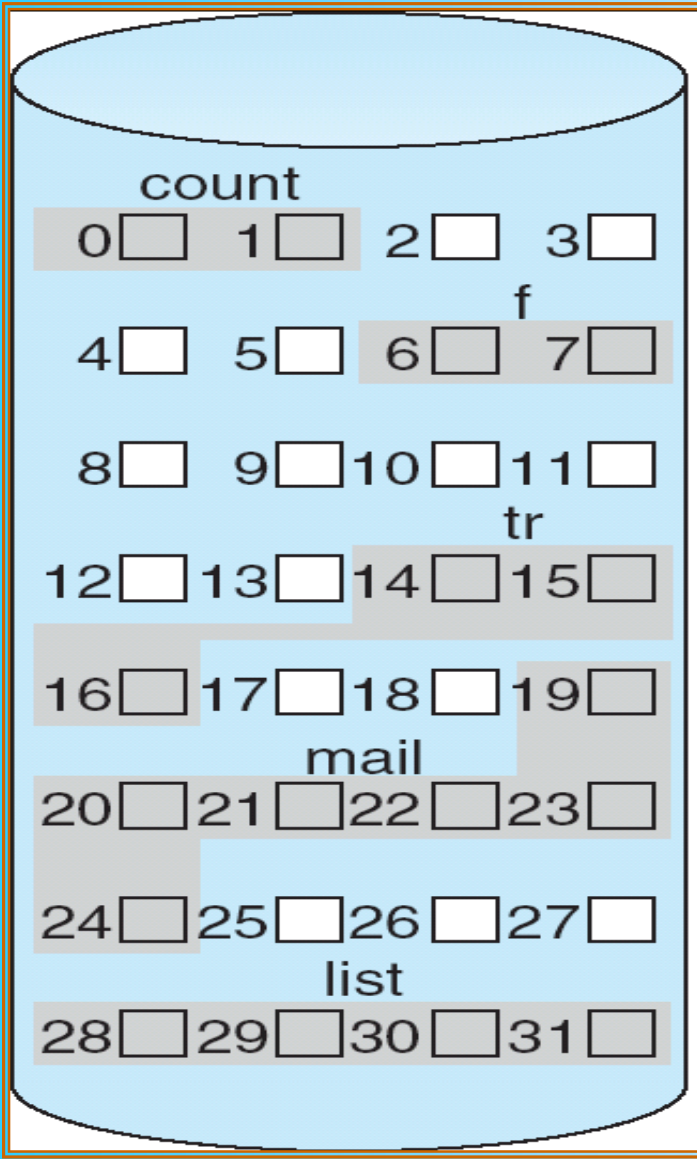


File Allocation Table

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

- Best method for sequential file
- Easy to retrieve a single block

Figure 12.8 Contiguous File Allocation (After Compaction)



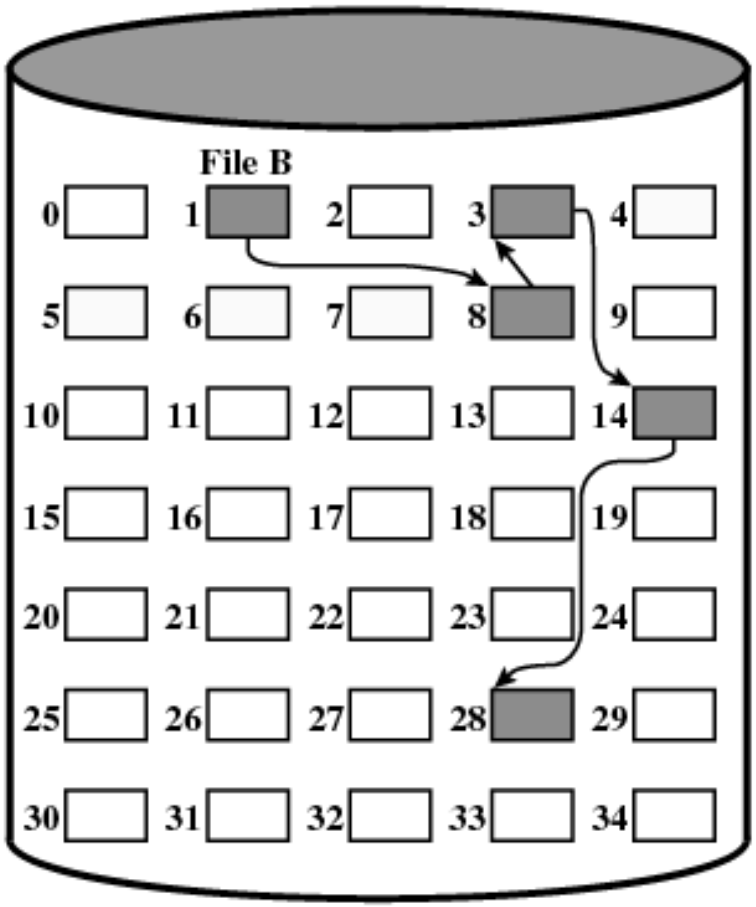
directory

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

Another example of contiguous allocation

Linked/Chained Allocation

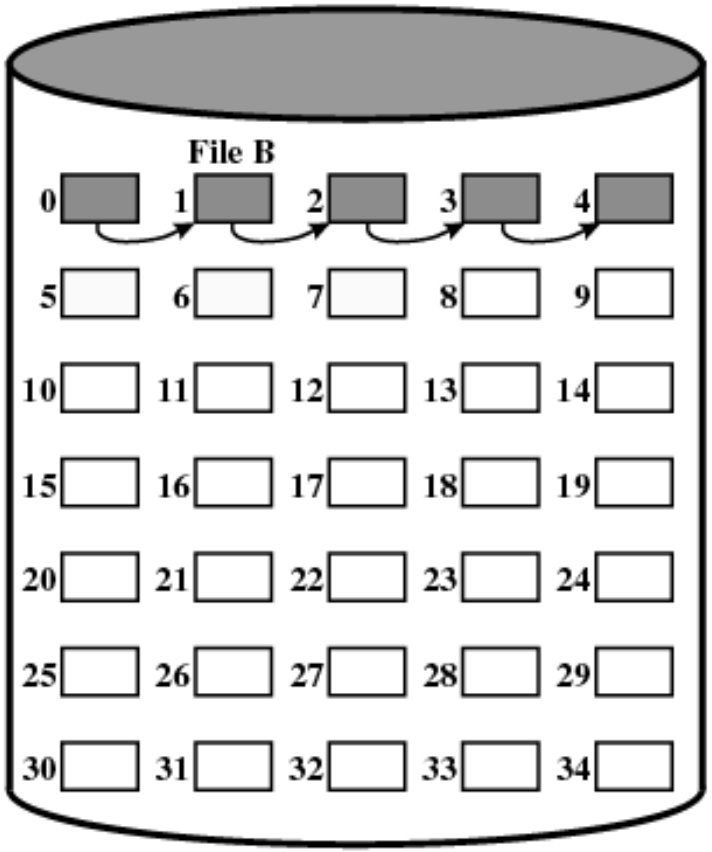
- Allocation on basis of individual block
- Each block contains a pointer to the next block in the chain
- Only single entry in the file allocation table
  - Starting block and length of file
- No external fragmentation
  - Any free block can be added to a chain
- Best for sequential files
- No accommodation of the principle of locality



File Allocation Table

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

Figure 12.9 Chained Allocation

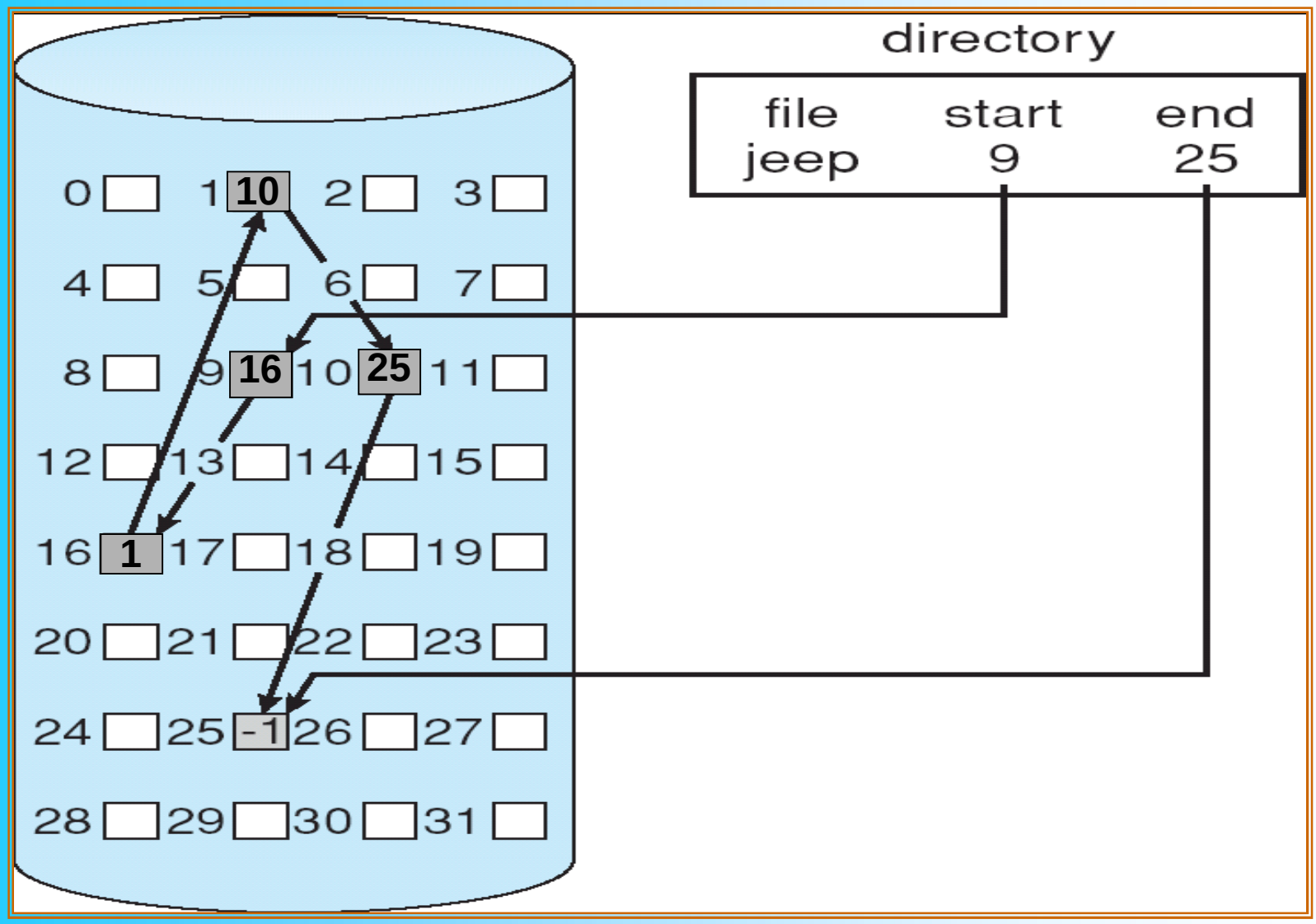


File Allocation Table

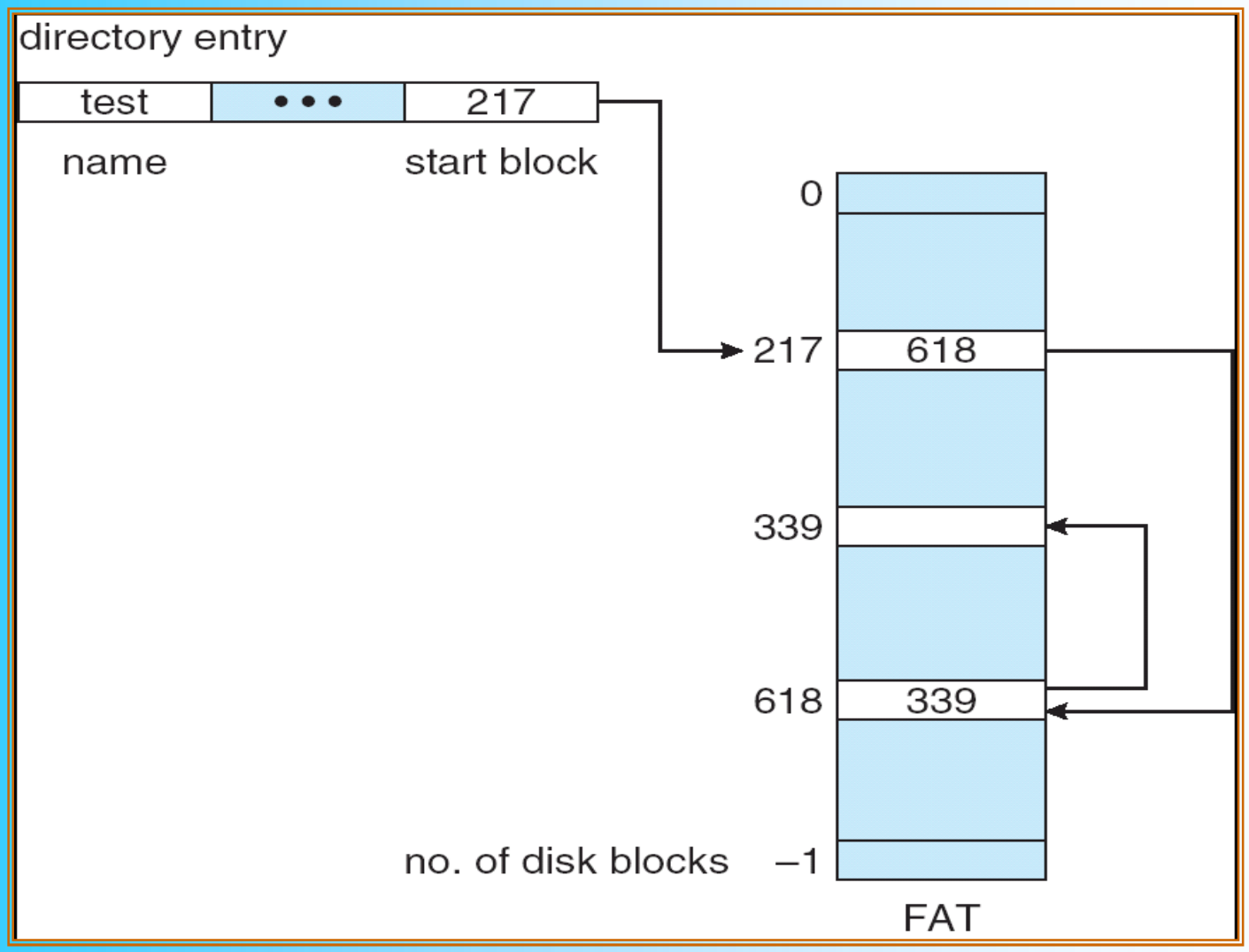
File Name	Start Block	Length
...	...	...
File B	0	5
...	...	...

Figure 12.10 Chained Allocation (After Consolidation)

Linked Allocation



File-Allocation Table



Indexed Allocation

- File allocation table contains a separate one-level index for each file
- The index has one entry for each portion allocated to the file
- The file allocation table contains block number for the index

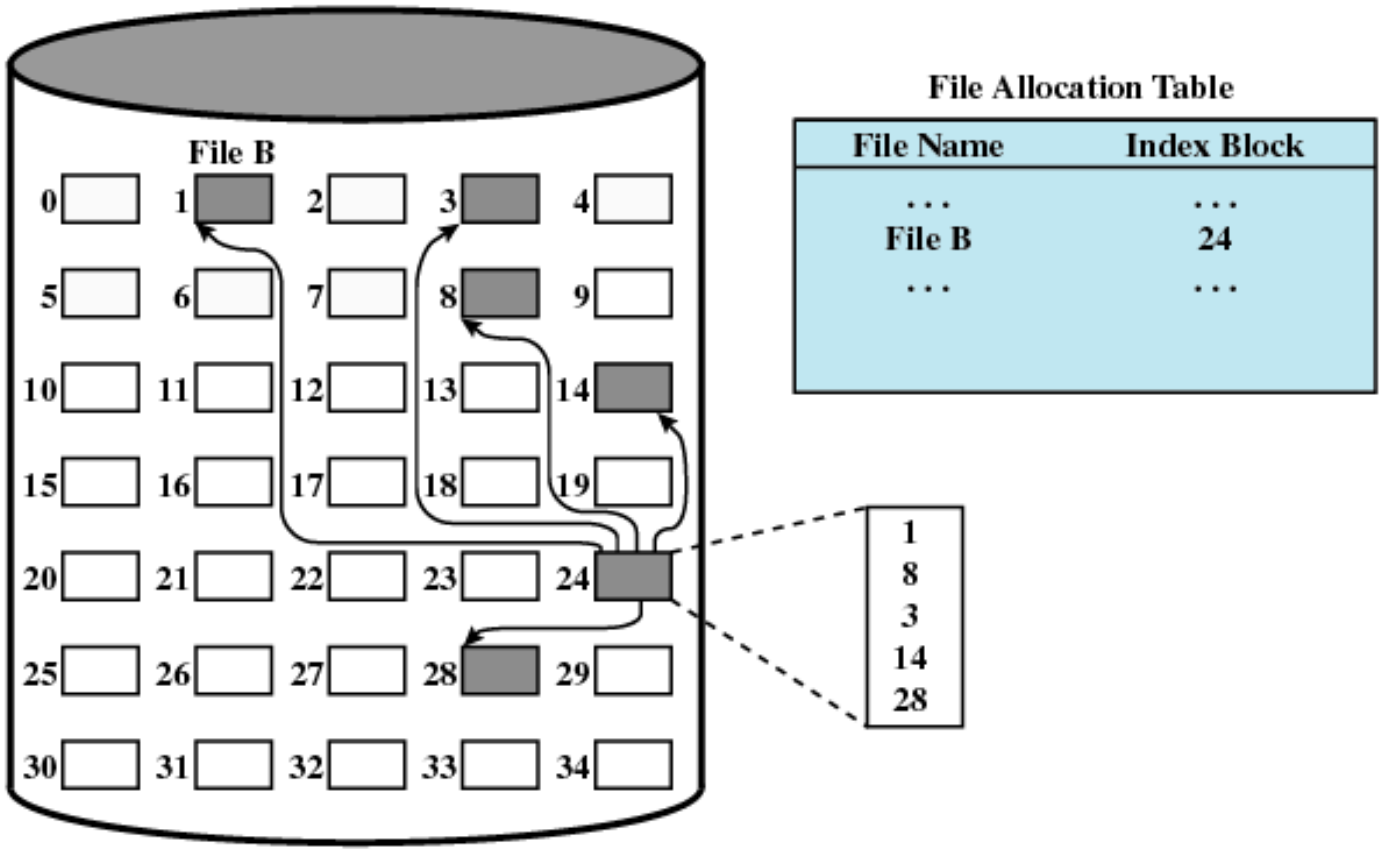


Figure 12.11 Indexed Allocation with Block Portions

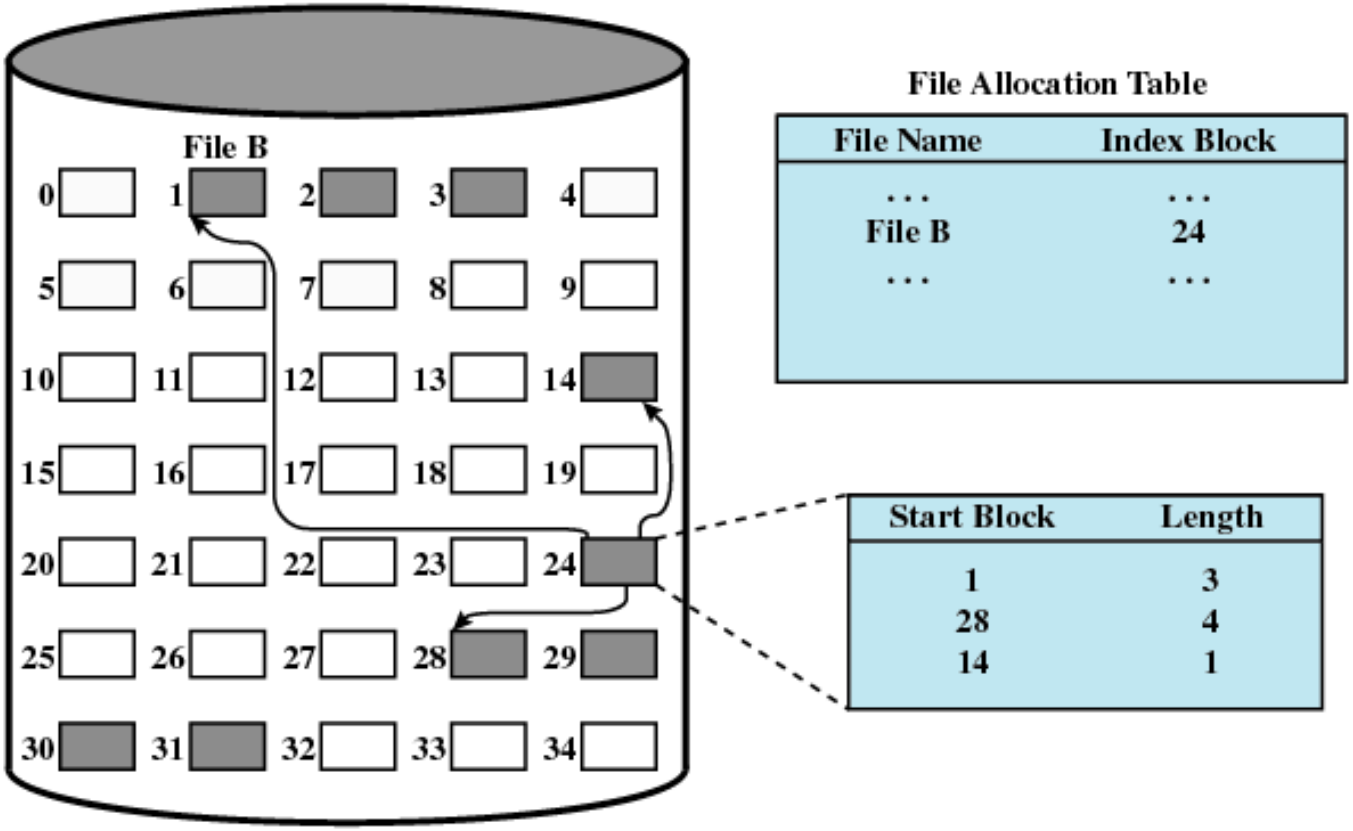
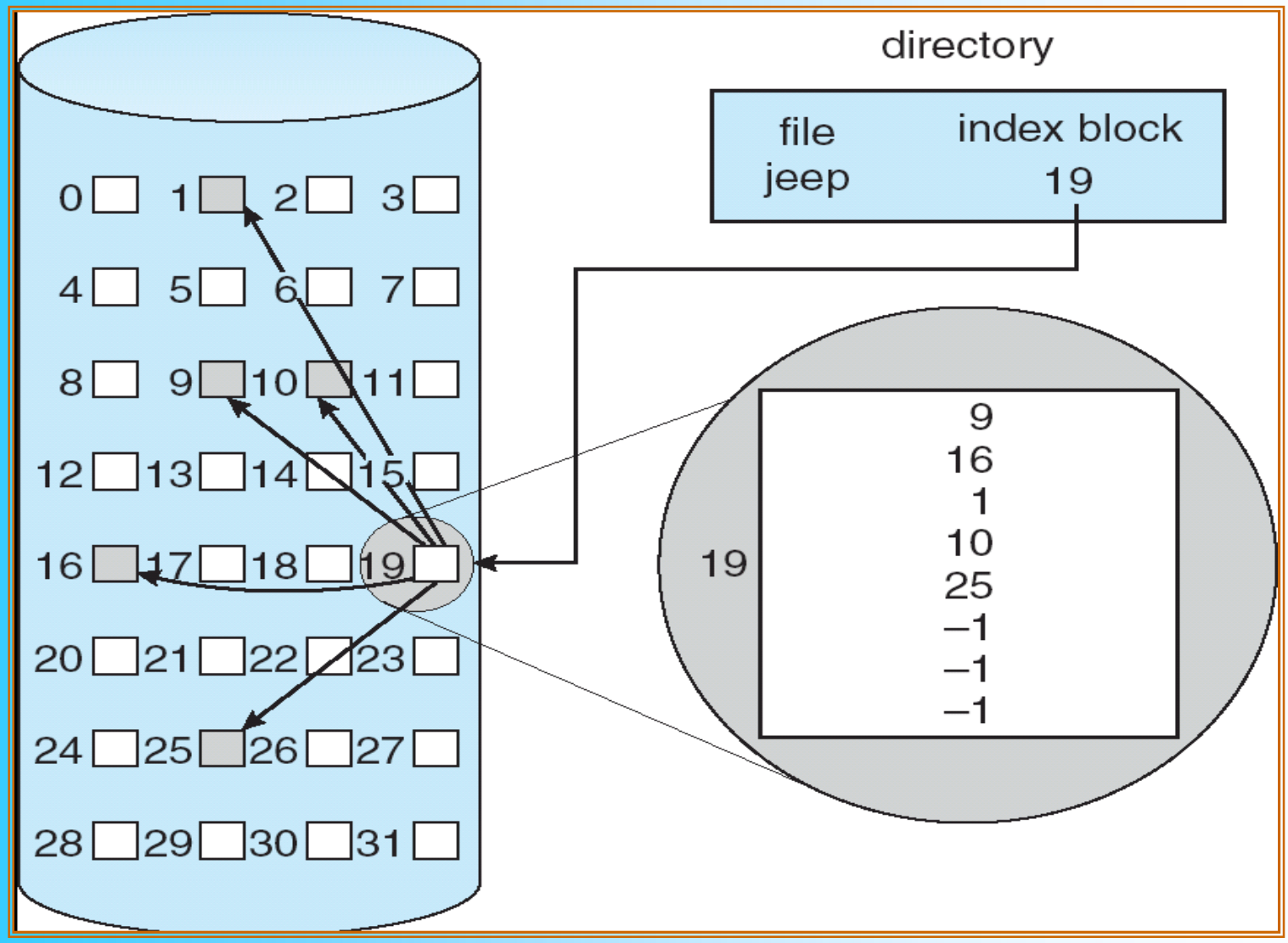


Figure 12.12 Indexed Allocation with Variable-Length Portions

Example of Indexed Allocation



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, other magnetic disk, optical)
- Recover lost file or disk by **restoring** data from backup