Chapter 12
File Management

(based on original slides by Pearson)

File System

- Reasons to have a file system:
  - Long-term storage
  - Sharable data between processes
  - Structured, hierarchic relationships among files

Typical File Operations

- Create
- Delete
- Open
- Close
- Read
- Write
- (Seek)

File Terminology

- Field
  - Basic element of data
  - Contains a single value (student name)
  - Characterized by its length and data type
- Record
  - Collection of related fields
  - Treated as a unit

File Terminology

- 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 Record Operations

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

File Management Systems (FMS)

… is a set of system software that provides services to users & applications in the use of files.

- Sole interface for the users & applications
- Management provided by the OS:
  - Programmer does not need to develop file management software

Objectives for a FMS

- Meet the data management needs and requirements of the user
- Guarantee that the data in the file are valid
- Optimize performance (user&system side)
- Provide I/O support for a variety of storage device types

Objectives for a FMS

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

Minimal Set of Requirements

- Authorized users can create, delete, read, write and modify files
- Authorized users have controlled access to other users' files
- Authorized users can control access permissions for own files
- Authorized users can restructure their files

Minimal Set of Requirements

- Authorized users can move data between files
- Authorized users can back up and recover their files in case of damage
- Authorized users can access their files by using symbolic names

Files System Software Architecture

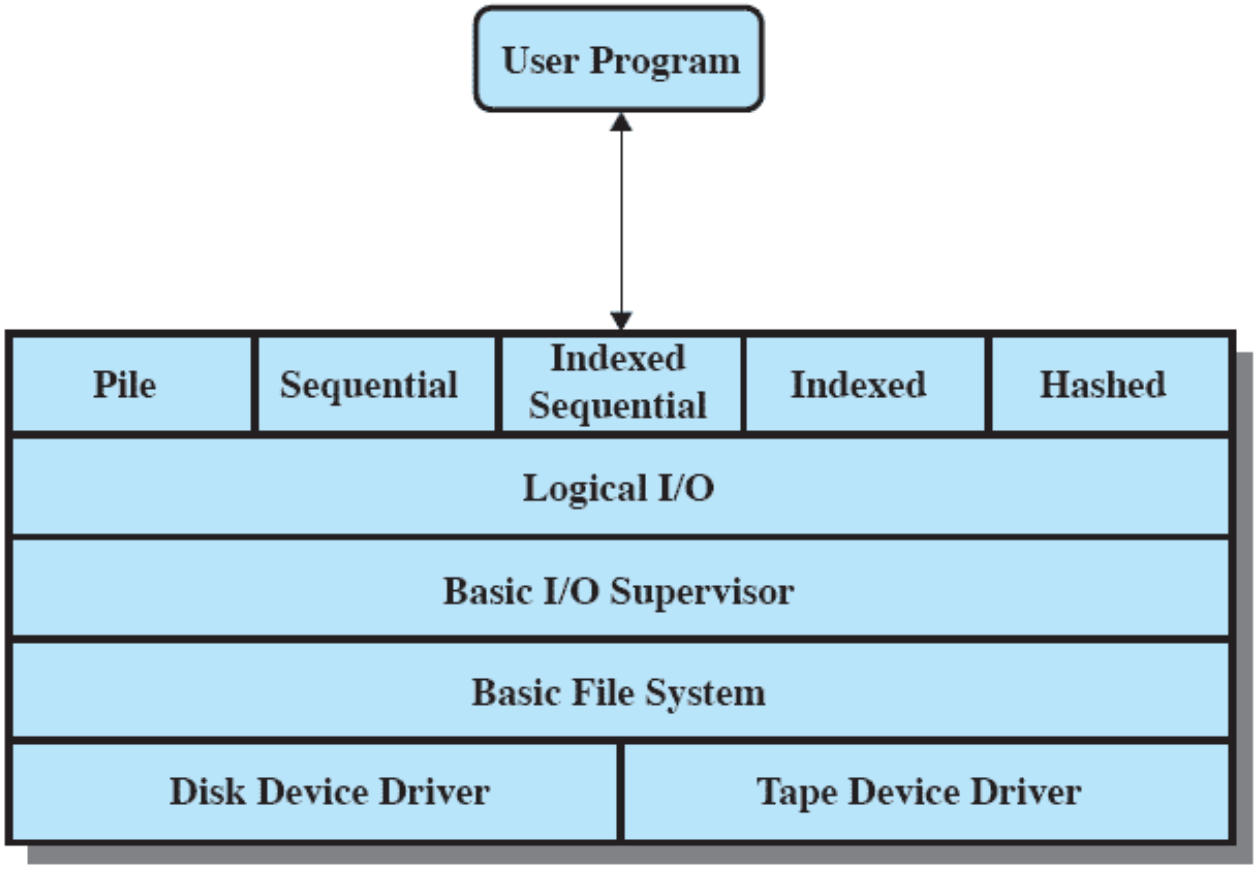


Figure 12.1 File System Software Architecture

Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- Responsible for starting I/O operations on a device
- Processes the completion of an I/O request

Basic File System

- Also called 'Physical I/O'
- Does not understand data, only handles it
  - Deals with exchanging blocks of data
  - Concerned with the placement of blocks
  - Concerned with buffering blocks in main memory

Basic I/O Supervisor

- Maintains control structure for device I/O, scheduling, and file status
- Optimizes access to device I/O
- Manages I/O buffers

Logical I/O

- Enables users and applications to access records
- Provides general-purpose record I/O capability
- Maintains basic data about file

Access Method

- Reflect different file structures
- Different ways to access and process data
- Can optimize access depending on the application needs:
  - Sequential access (one after the other)
  - Direct access (direct address)
  - Indexed access (use an identifier)

Elements of File Management

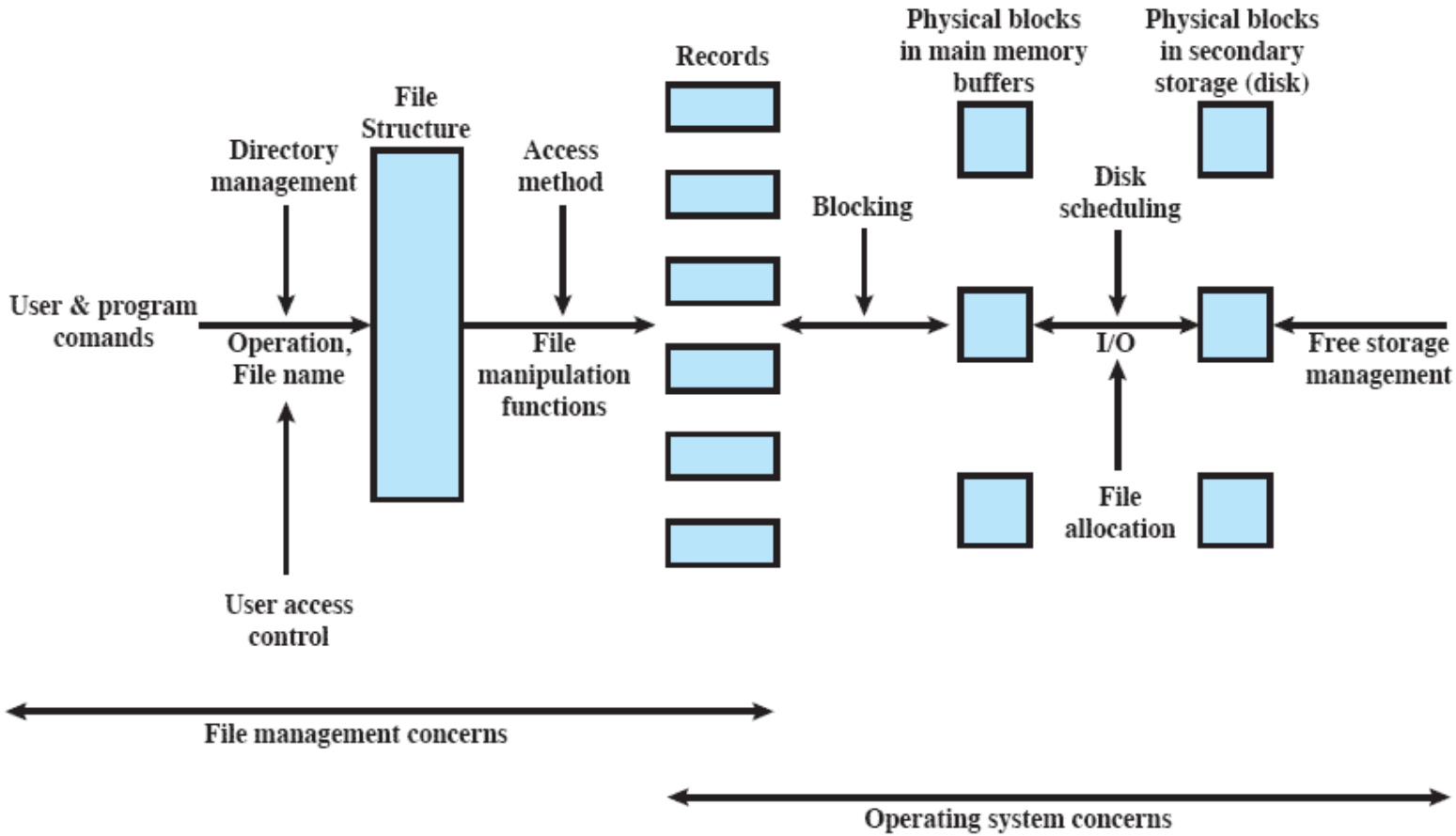


Figure 12.2 Elements of File Management

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

File Organization

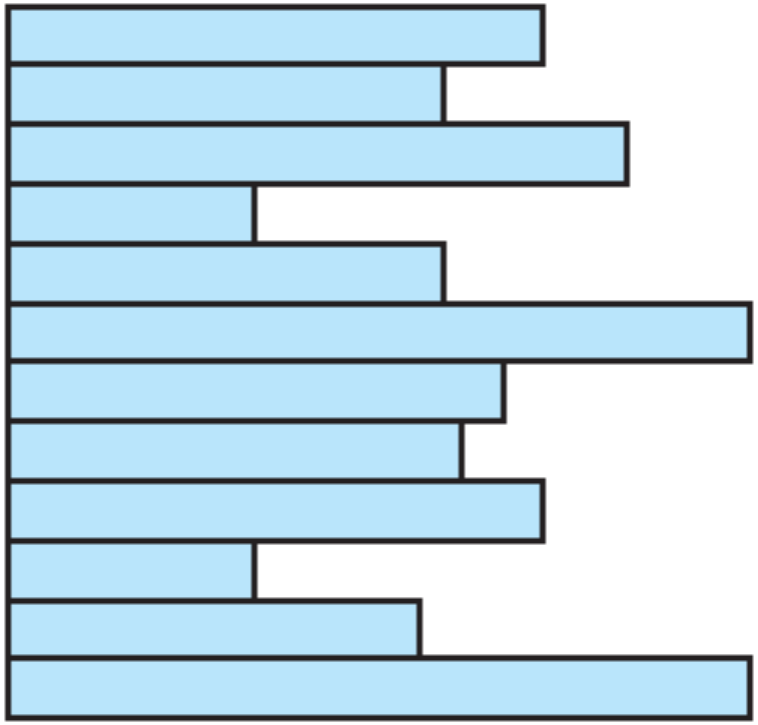
… logical structuring of records

- Criteria:
  - Short access time
    - Needed when accessing a single record
  - Ease of update
    - File on CD-ROM will not be updated, so this is not a concern

Criteria (ctnd.)

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

The Pile



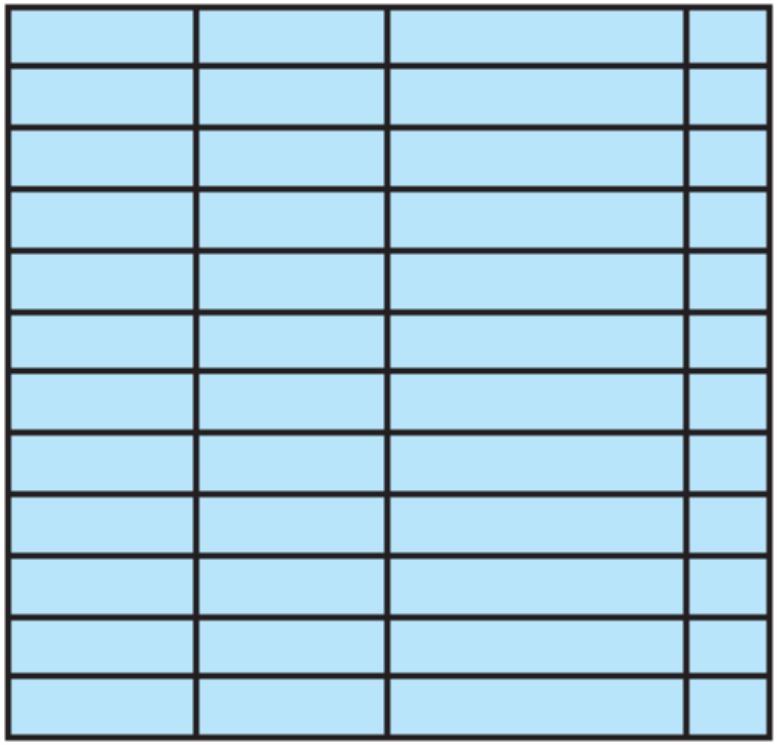
Variable-length records
Variable set of fields
Chronological order

(a) Pile File

File Organization

- The Pile
  - 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
  - Useful when data is only collected and stored prior to processing

The Sequential File



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

(b) Sequential File

File Organization

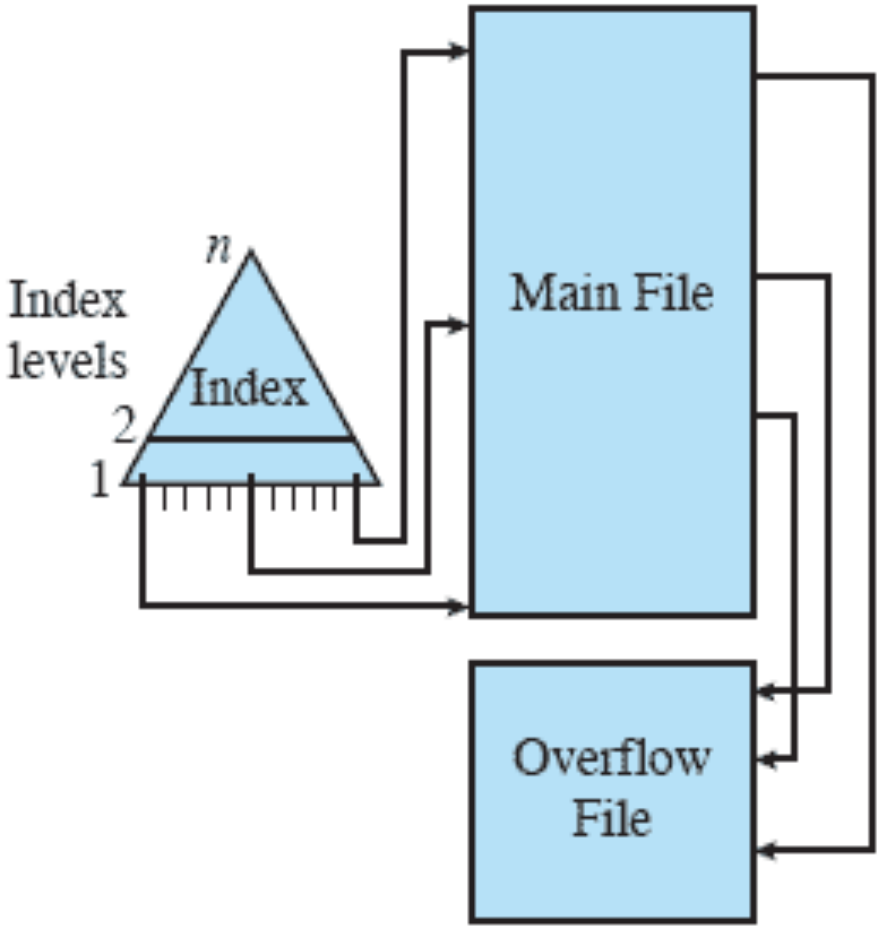
- The Sequential File
  - Fixed format used for records
  - Records are the same length
  - All fields the same (order and length)
  - Field names and lengths are attributes of the file

File Organization

- The Sequential File
  - One field is the key filed
    - Uniquely identifies the record
    - Records are stored in key sequence
  - Search is inefficient (look through the whole file)

- Updates are a pain, because physical organization matches the logical organization
  - Overflow pile, transaction log & periodic batch updates to the file that rearranges the file
  - Or: different physical layout from logical layout

Indexed Sequential File



(c) Indexed Sequential File

File Organization

- 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

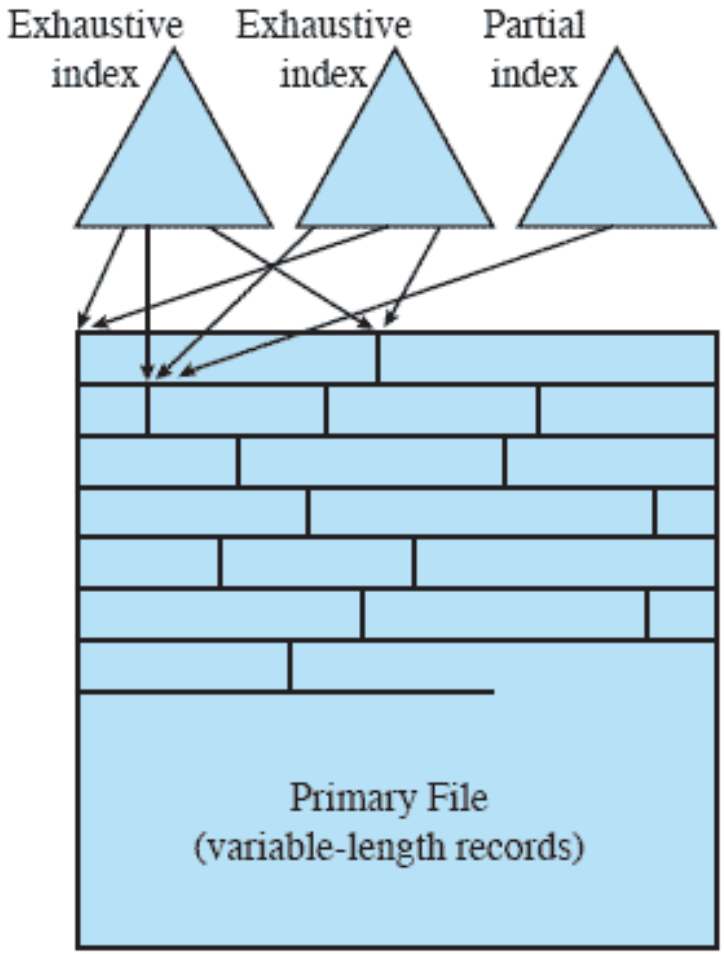
File Organization

- Comparison of sequential and indexed sequential
  - Example: a file contains 1 million records
  - On average 500,000 accesses are required to find a record in a sequential file
  - 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

File Organization

- 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
  - Upon NULL pointer, continue in the main file
  - 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 File



(d) Indexed File

File Organization

- 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
  - May contain a partial index

File Organization

- The Direct or Hashed File
  - Directly access a block at a known address
  - Key field required for each record

Performance

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 Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- Directory itself is a file usually accessed indirectly by the user
- Provides mapping between file names and the files themselves

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

Information Elements of a File Directory

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

Information Elements of a File Directory

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

Single List Directory

- Approach I: single list
  - 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

Two-Level Directory

- Approach II: **one list per user**
  - 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 (unique naming, no group organization [project files, …])

Hierarchical, or Tree-Structured Directory

- Approach III: **tree structure**
  - Master directory with user directories underneath it
  - Each user directory may have subdirectories and files as entries

Tree-Structured Directory

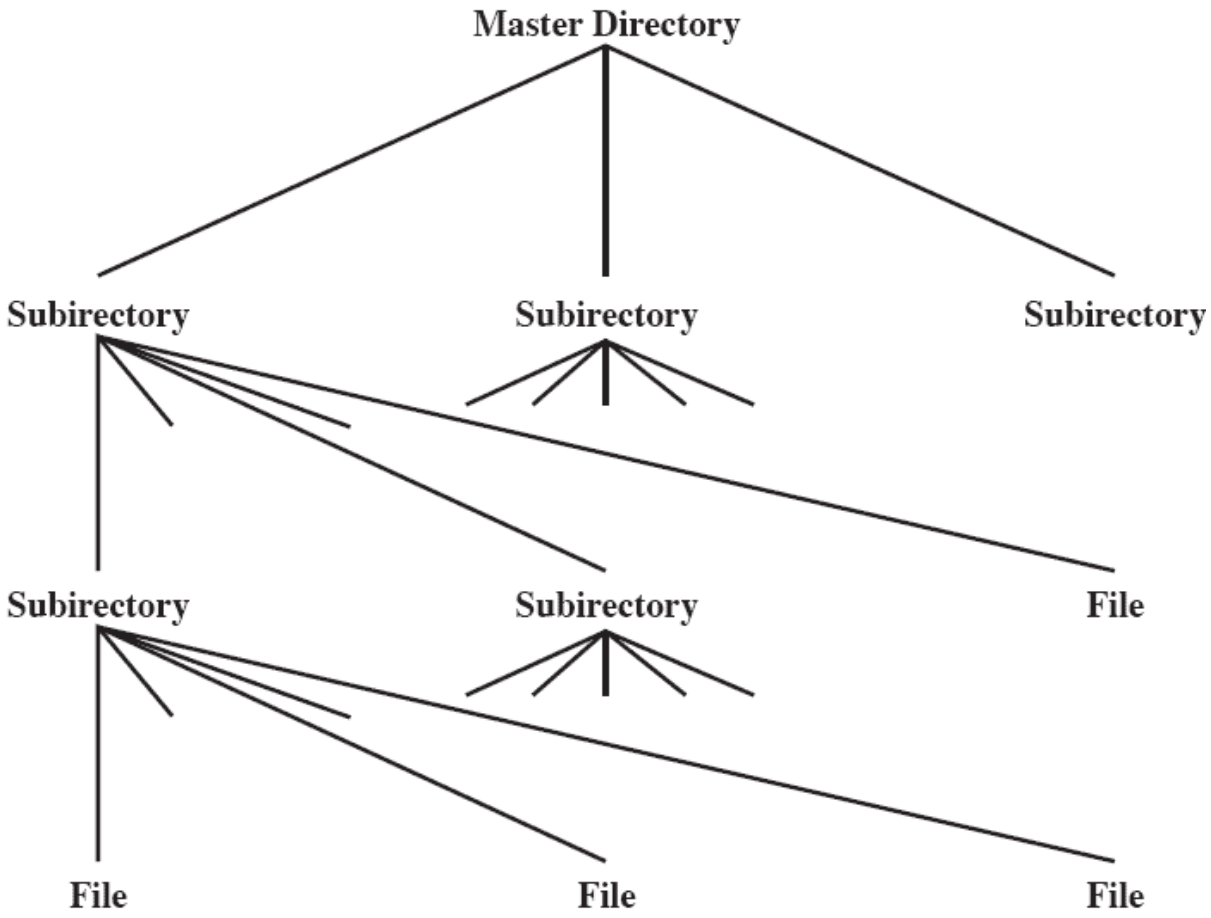


Figure 12.4 Tree-Structured Directory

Example of Tree-Structured Directory

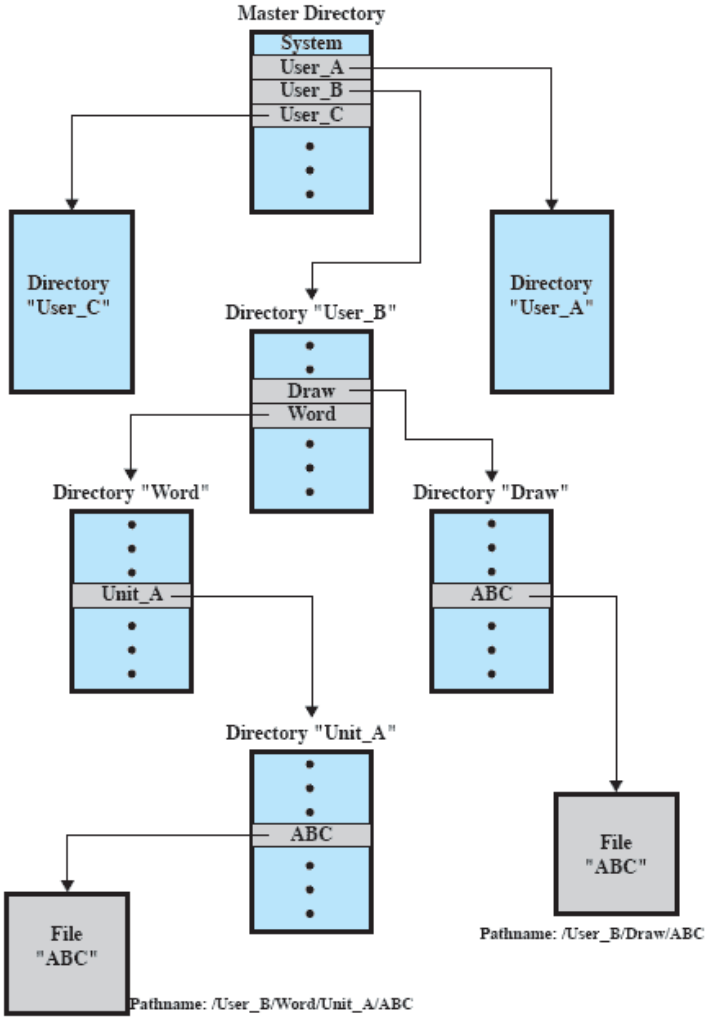


Figure 12.5 Example of Tree-Structured Directory

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

File Sharing

- In multiuser system, allow files to be shared among users
- Two issues
  - Access rights
  - Management of simultaneous access

Access Rights

- None
  - User may not know of the existence of the file
  - 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

Access Rights

- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- 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, deleted, 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
- Mutual exclusion and deadlock are issues for shared access

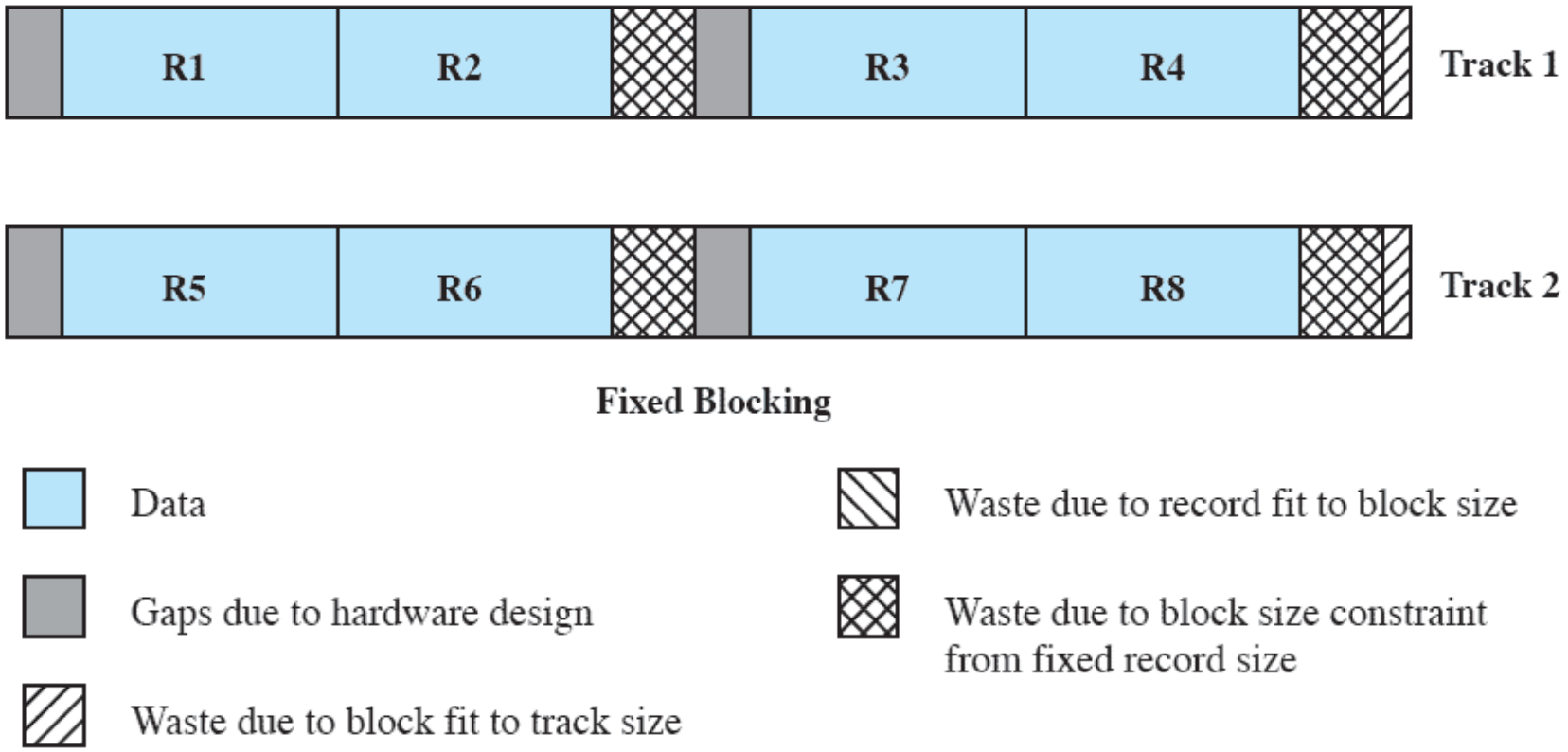
Record Blocking

- Organize records in blocks on the I/O device

- Questions:
  - Variable or fixed block size?
  - Relative bock size to average record size?

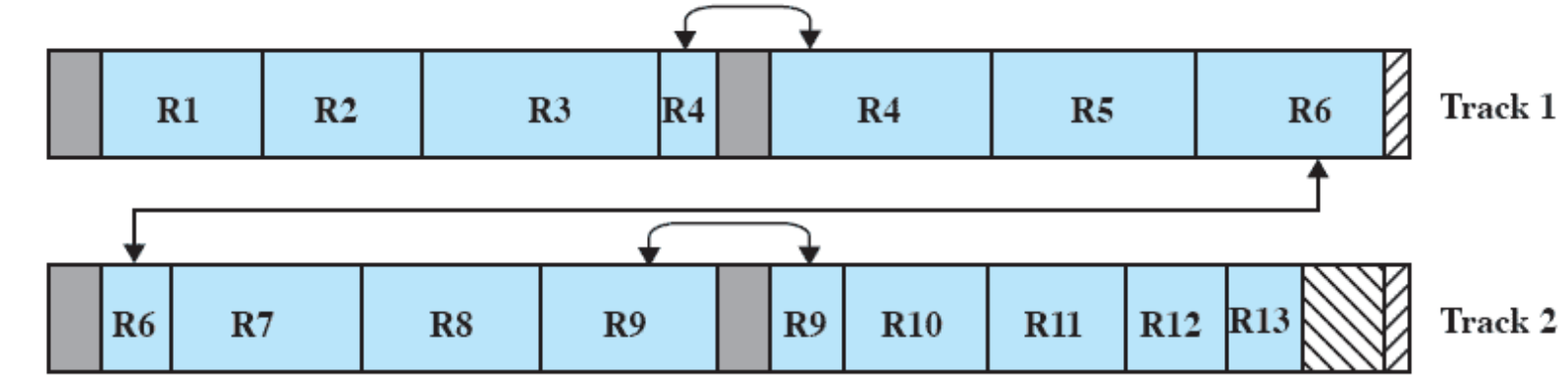
Fixed Blocking

- Fixed-length records, storing integral number in a block



Variable Blocking: Spanned

- Variable-length records, some records span two blocks with a continuation pointer



Variable Blocking: Spanned

Data

Gaps due to hardware design

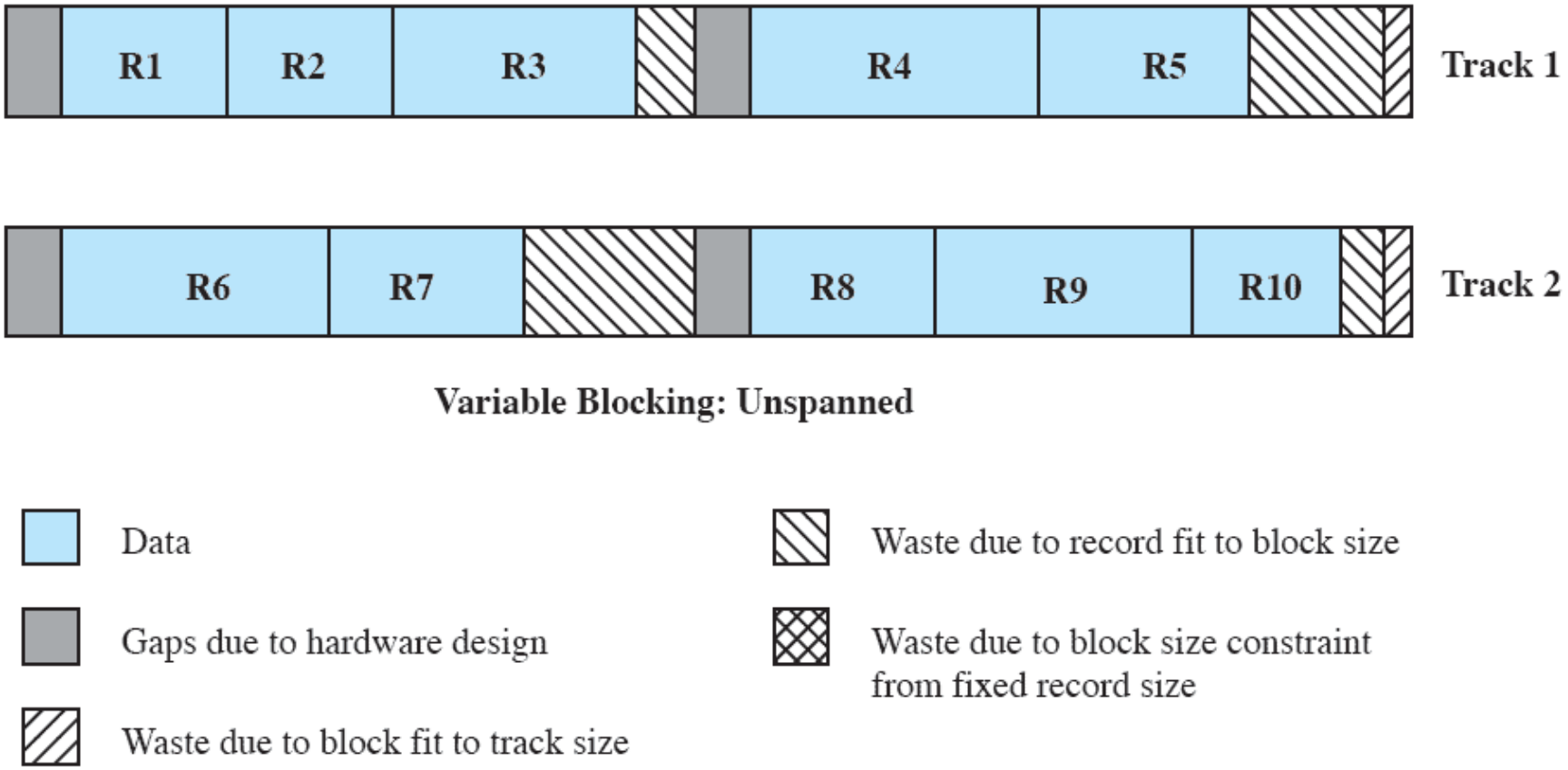
Waste due to block fit to track size

Waste due to record fit to block size

Waste due to block size constraint from fixed record size

Variable Blocking: Unspanned

- Variable-length records, no spanning



Secondary Storage Management

- Space must be allocated to files
- Must keep track of the space available for allocation

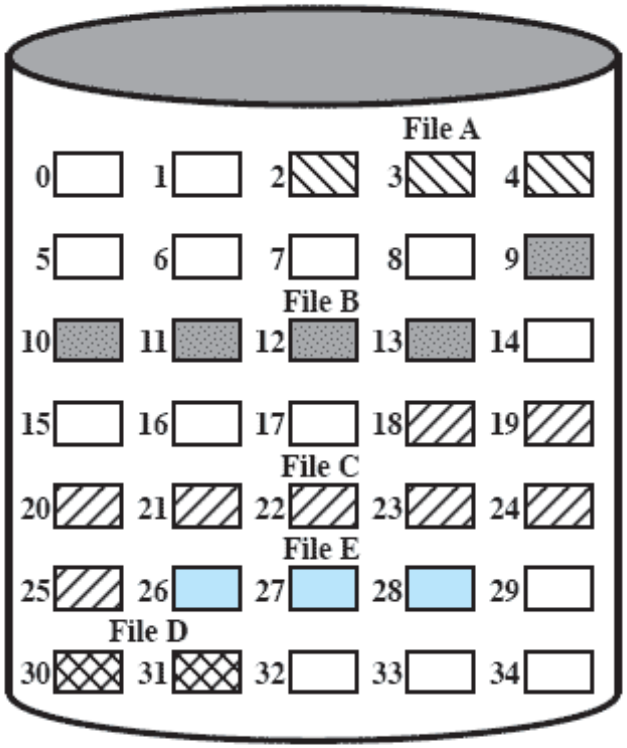
Preallocation

- 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 overestimate file size so as not to run out of space

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

Contiguous File Allocation

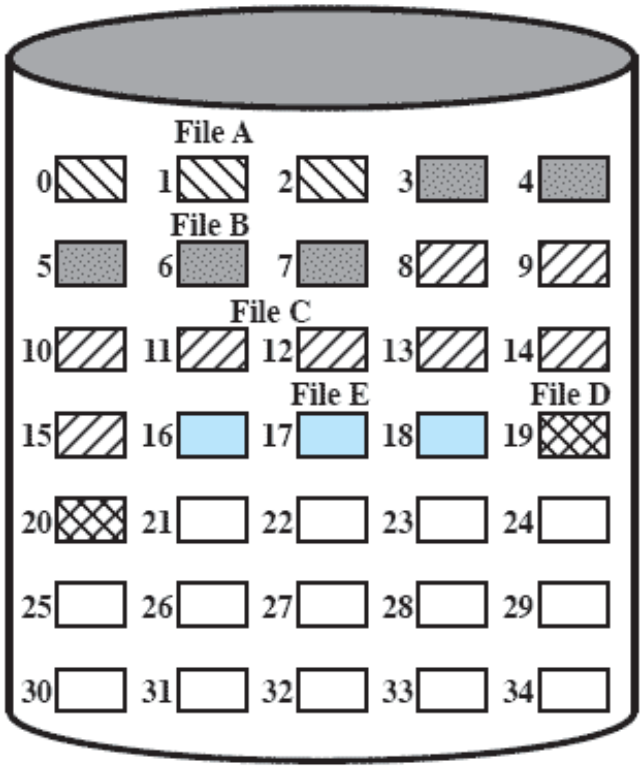


File Allocation Table

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

Figure 12.7 Contiguous File Allocation

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

Figure 12.8 Contiguous File Allocation (After Compaction)

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

Chained Allocation

- No external fragmentation
- Best for sequential files
- No accommodation of the principle of locality

Chained Allocation

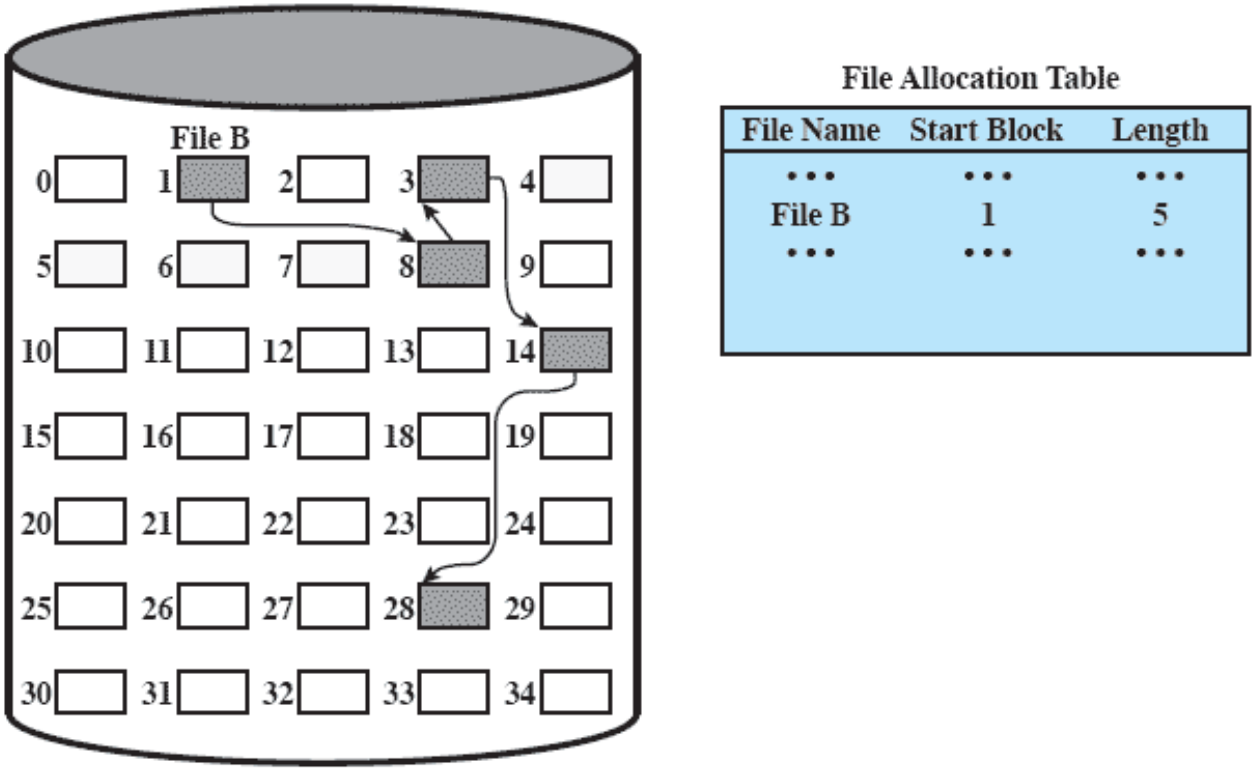
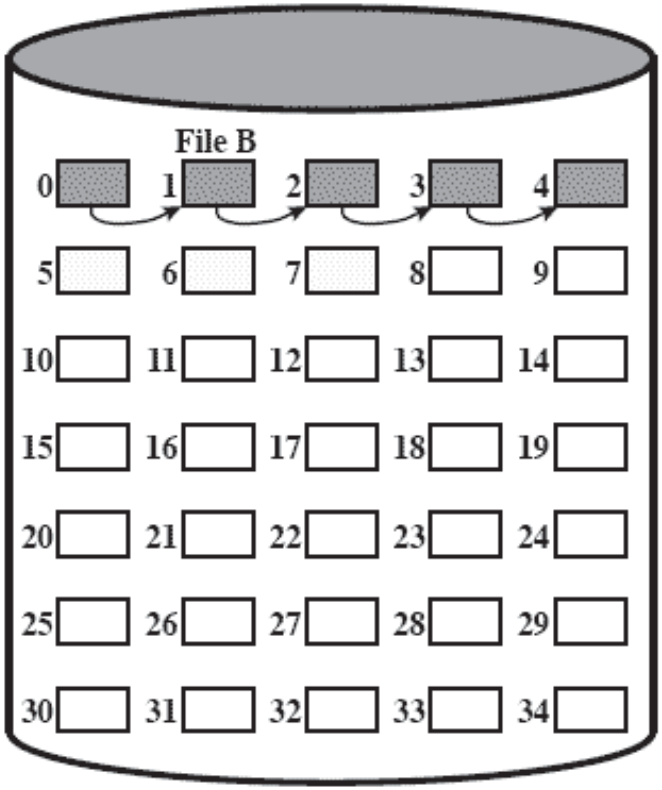


Figure 12.9 Chained Allocation

Chained Allocation



File Allocation Table

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

Figure 12.10 Chained Allocation (After Consolidation)

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

Indexed Allocation

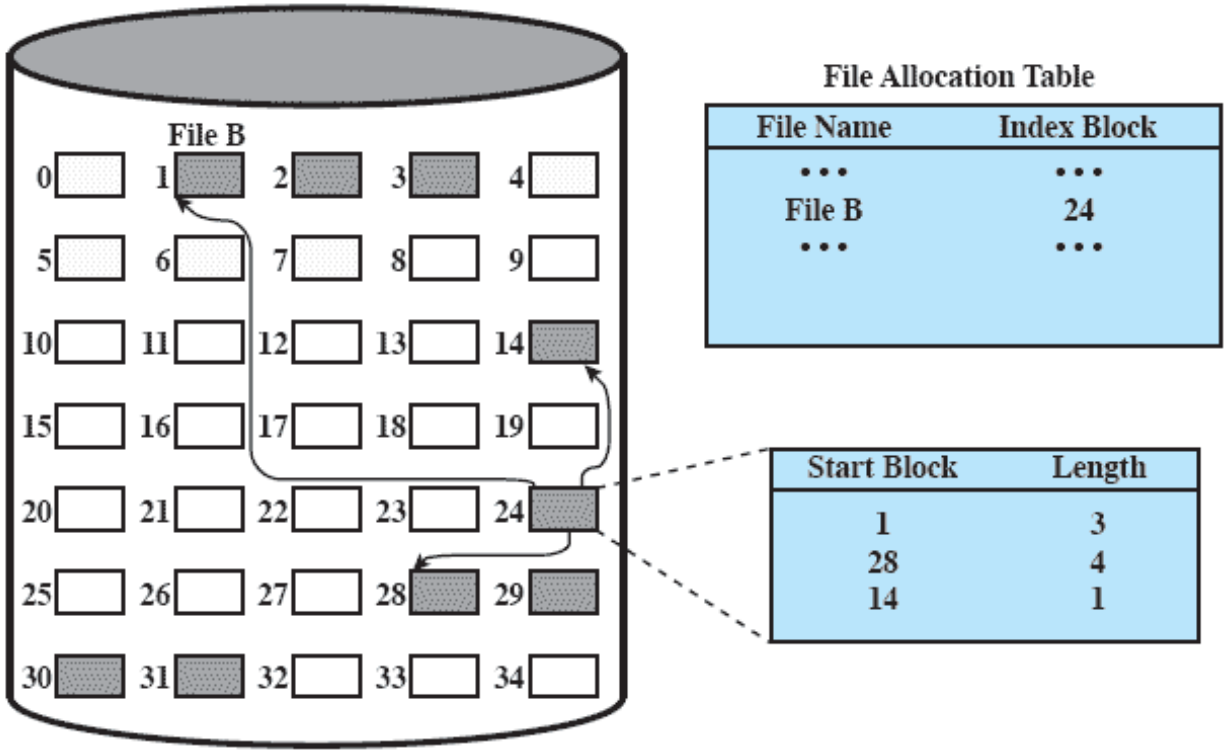


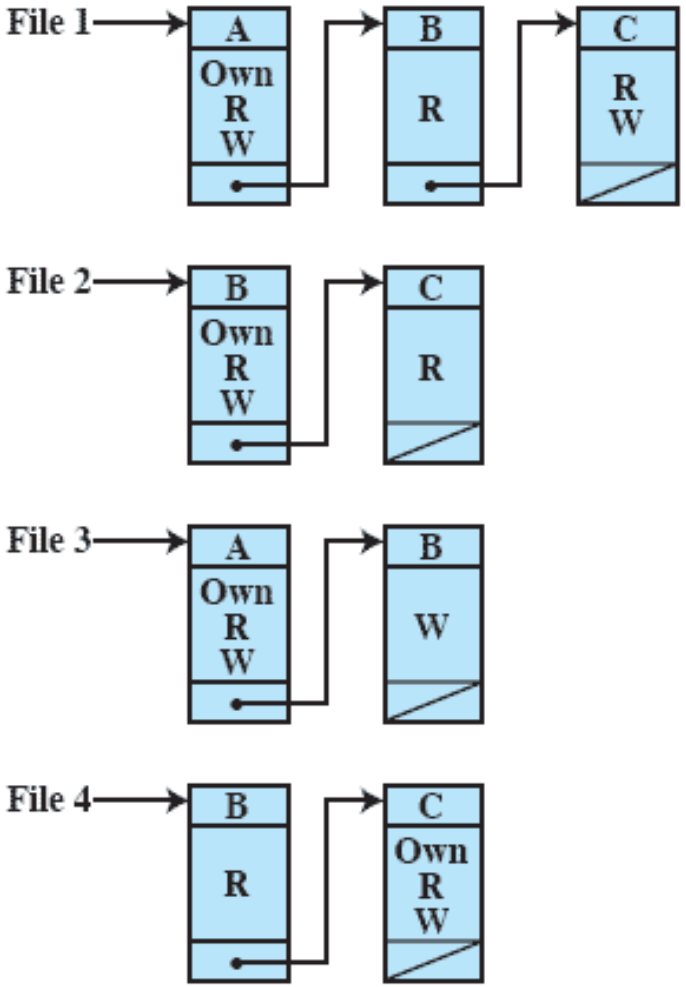
Figure 12.12 Indexed Allocation with Variable-Length Portions

Access Matrix

	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

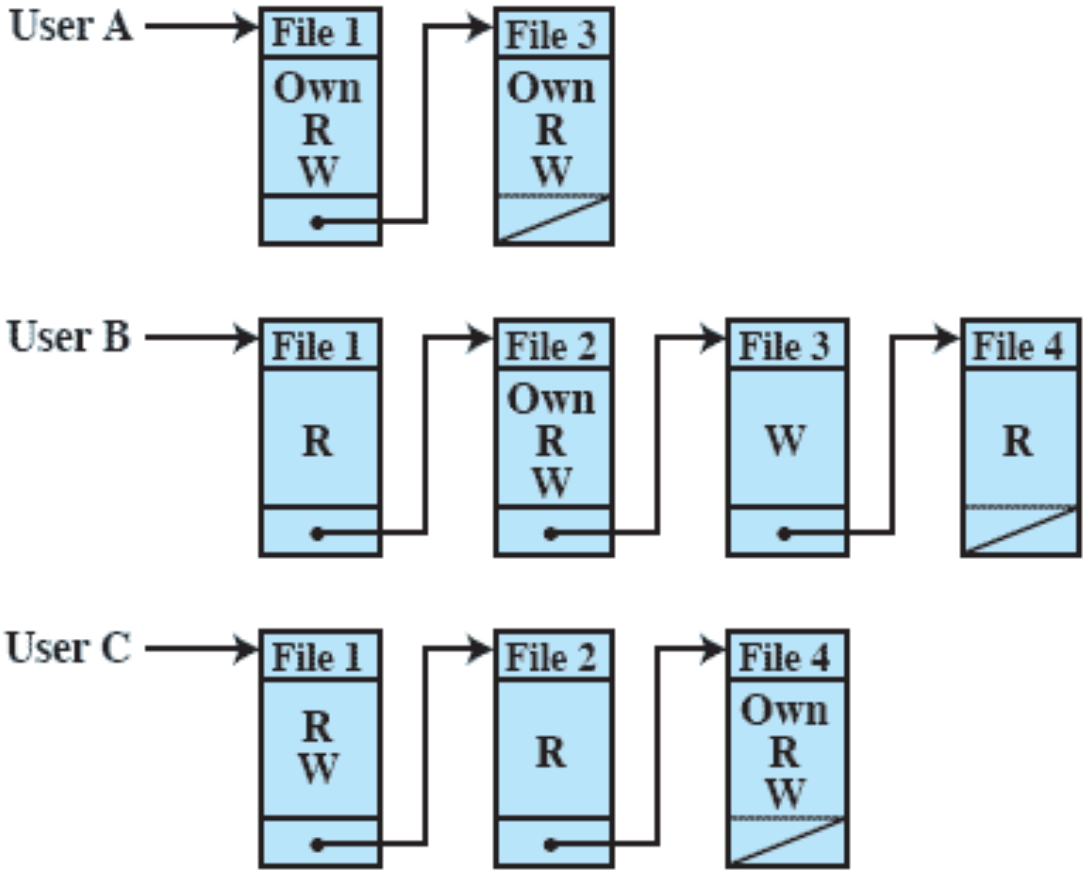
(a) Access matrix

Access Control List



(b) Access control lists for files of part (a)

Capability Lists



(c) Capability lists for files of part (a)