Understanding Operating Systems Fifth Edition

Chapter 8
File Management

Learning Objectives

- The fundamentals of file management and the structure of the file management system
- File-naming conventions, including the role of extensions
- The difference between fixed-length and variable-length record format
- The advantages and disadvantages of contiguous, noncontiguous, and indexed file storage techniques

Learning Objectives (continued)

- Comparisons of sequential and direct file access
- The security ramifications of access control techniques and how they compare
- The role of data compression in file storage

The File Manager

- **File management system**
  - Software
- File access responsibilities
  - Creating, deleting, modifying, controlling
- Support for libraries of programs
  - Online users
    - Spooling operations
    - Interactive computing
- Collaborates with device manager

Responsibilities of the File Manager

- **Four tasks**
  - File storage tracking
  - Policy implementation
    - Determine where and how files are stored
    - Efficiently use available storage space
    - Provide efficient file access
  - File allocation if user access cleared
    - Record file use
  - File deallocation
    - File returned to storage
    - Communicate file availability

Responsibilities of the File Manager (continued)

- Policy determines:
  - File storage location
  - System and user access
    - Uses device-independent commands
- Access to material
  - Two factors
- Flexibility of access to information (Factor 1)
  - Shared files
  - Providing distributed access
  - Allowing users to browse public directories

Responsibilities of the File Manager (continued)

- Subsequent protection (Factor 2)
  - Prevent system malfunctions
  - Security checks
    - Account numbers, passwords, lockwords
- **File allocation**
  - Activate secondary storage device, load file into memory, update records
- **File deallocation**
  - Update file tables, rewrite file (if revised), notify waiting processes of file availability

Definitions

- **Field**
  - Group of related bytes
  - Identified by user (name, type, size)
- **Record**
  - Group of related fields
- **File**
  - Group of related records
  - Information used by specific application programs
    - Report generation
  - Flat file
    - No connections to other files, no dimensionality

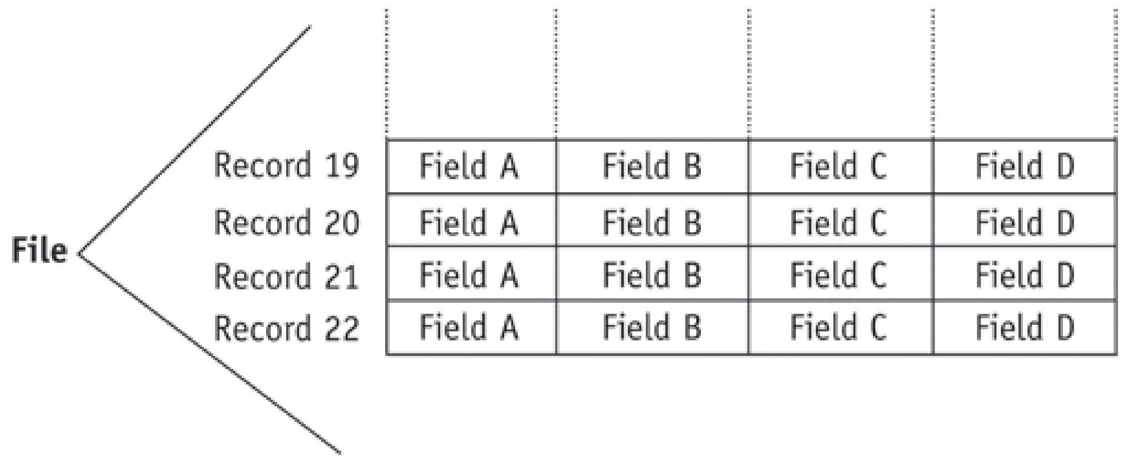
Definitions (continued)

- **Databases**
  - Groups of related files
  - Interconnected at various levels
    - Give users flexibility of access to stored data
- **Program files**
  - Contain instructions
- **Data files**
  - Contain data
- **Directories**
  - Listings of filenames and their attributes

Definitions (continued)

(figure 8.1)

Files are made up of records. Records consist of fields.



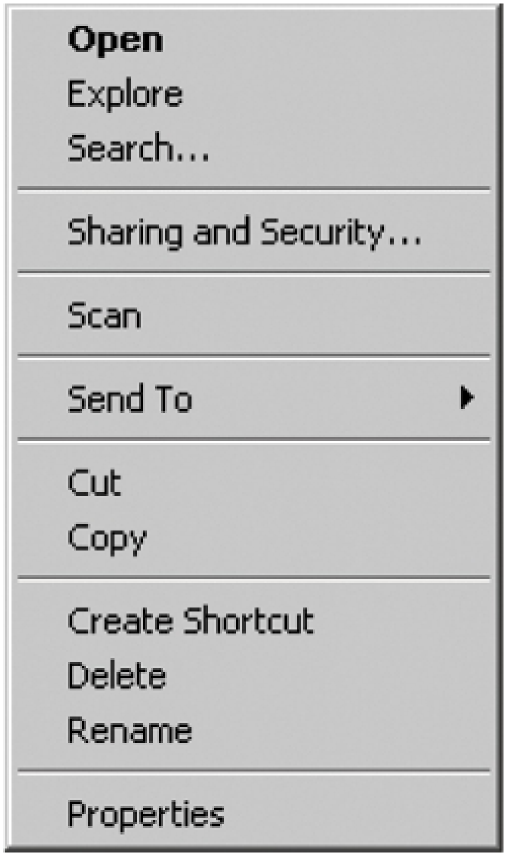
Interacting with the File Manager

- Commands
  - **Embedded in program**
    - OPEN, CLOSE, READ, WRITE, MODIFY
  - **Submitted interactively**
    - CREATE, DELETE, RENAME, COPY
- **Device independent**
  - Physical location knowledge not needed
    - Cylinder, surface, sector
  - Device medium knowledge not needed
    - Tape, magnetic disk, optical disc, flash storage
  - Network knowledge not needed

Interacting with the File Manager (continued)

(figure 8.2)

Typical menu of file options.



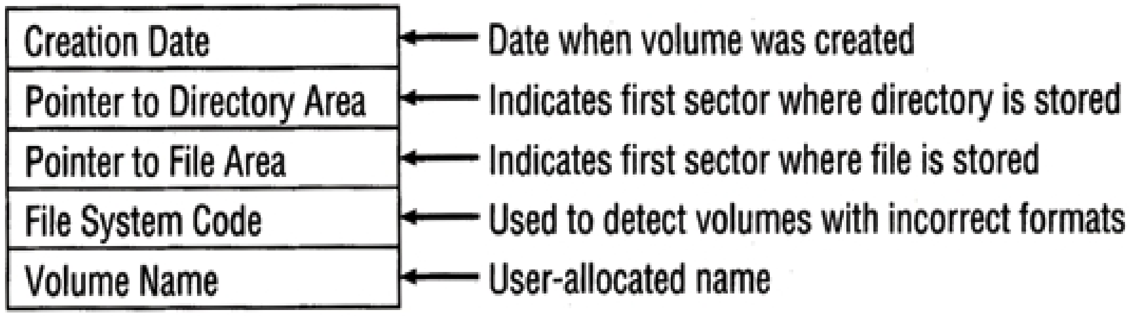
Interacting with the File Manager (continued)

- Logical commands
  - Broken into lower-level signals
  - Example: READ
    - Move read/write heads to record cylinder
    - Wait for rotational delay (sector containing record passes under read/write head)
    - Activate appropriate read/write head and read record
    - Transfer record to main memory
    - Send flag indicating free device for another request
- Performs error checking and correction
  - No need for error-checking code in programs

Typical Volume Configuration

- **Volume**
  - Secondary storage unit (removable, nonremovable)
  - Multifile volume
    - Contains many files
  - Multivolume files
    - Extremely large files spread across several volumes
- Volume name
  - File manager manages
  - Easily accessible
    - Innermost part of CD, beginning of tape, first sector of outermost track

Typical Volume Configuration (continued)


Creation Date
Pointer to Directory Area
Pointer to File Area
File System Code
Volume Name
Date when volume was created
Indicates first sector where directory is stored
Indicates first sector where file is stored
Used to detect volumes with incorrect formats
User-allocated name

(figure 8.3)

The volume descriptor, stored at the beginning of each volume, includes the volume name and other vital information about the storage unit.

Typical Volume Configuration (continued)

- **Master file directory (MFD)**
- Stored immediately after volume descriptor
- Lists
  - Names and characteristics of every file in volume
    - File names (program files, data files, system files)
  - Subdirectories
    - If supported by file manager
  - Remainder of volume
    - Used for file storage

Typical Volume Configuration (continued)

- **Single directory per volume**
  - Supported by early operating systems
- Disadvantages
  - Long search time for individual file
  - Directory space filled before disk storage space filled
  - Users cannot create subdirectories
  - Users cannot safeguard their files
  - Each program needs unique name
    - Even those serving many users

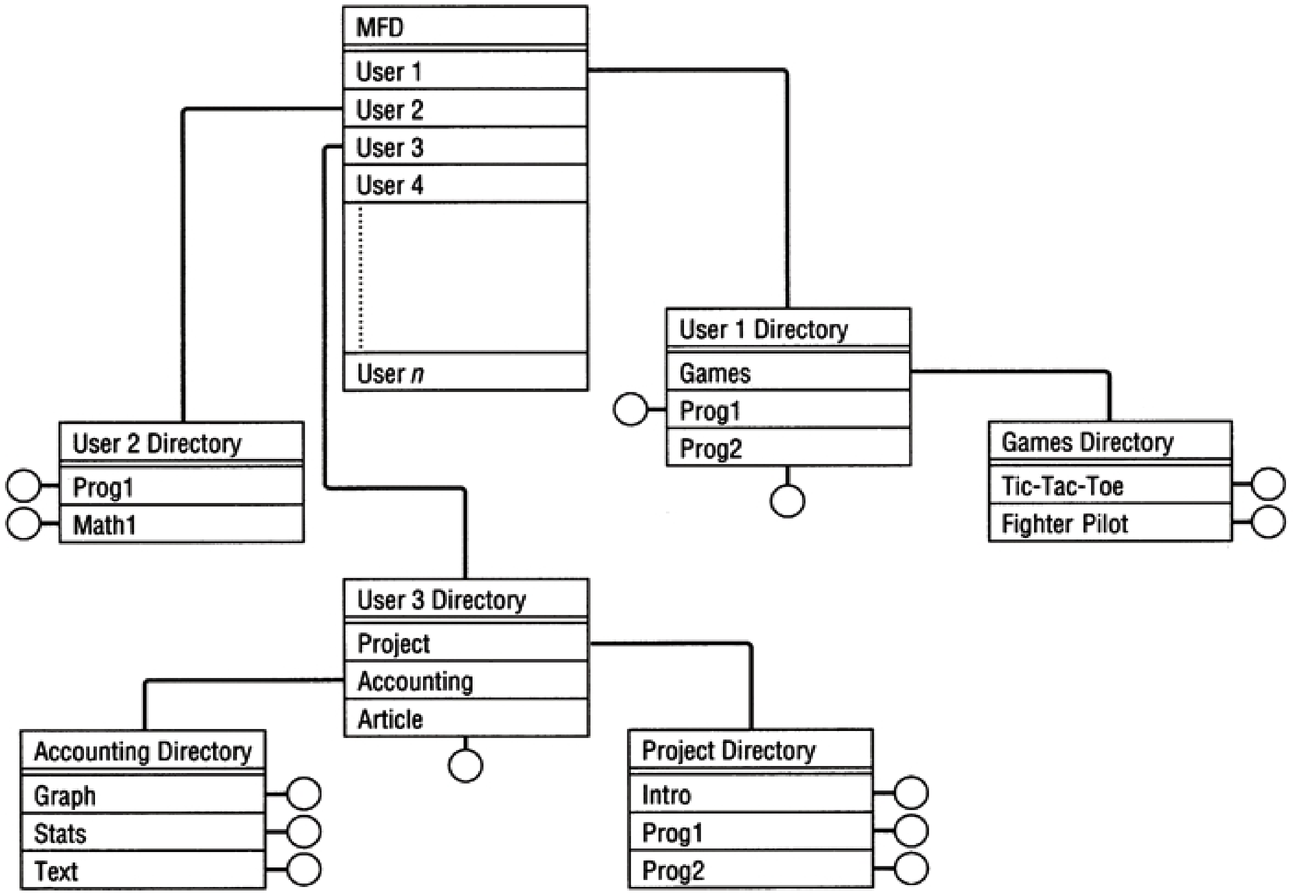
About Subdirectories

- **Newer file managers**
  - Create MFD for each volume
    - Contains file and subdirectory entries
  - Improvement over single directory scheme
    - Problems remain: unable to logically group files
- **Subdirectory**
  - Created upon account opening
  - Treated as file
    - Flagged in MFD as subdirectory
    - Unique properties

About Subdirectories (continued)

- File managers today
  - Users create own subdirectories (folders)
    - Related files grouped together
  - Implemented as upside-down tree
    - Efficient system searching of individual directories
    - May require several directories to reach file

About Subdirectories (continued)



(figure 8.4)

File directory tree structure. The "root" is the MFD shown at the top, each node is a directory file, and each branch is a directory entry pointing to either another directory or to a real file. All program and data files subsequently added to the tree are the leaves, represented by circles.

About Subdirectories (continued)

- **File descriptor**
  - Filename: ASCII code
  - File type: organization and usage
    - System dependent
  - File size: for convenience
  - File location
    - First physical block identification
  - Date and time of creation
  - Owner
  - Protection information: access restrictions
  - Record size: fixed size, maximum size

File-Naming Conventions

- Filename components
  - Relative filename and extension
- **Complete filename (absolute filename)**
  - Includes all path information
- **Relative filename**
  - Name without path information
  - Appears in directory listings, folders
  - Provides filename differentiation within directory
  - Varies in length
    - One to many characters
    - Operating system specific

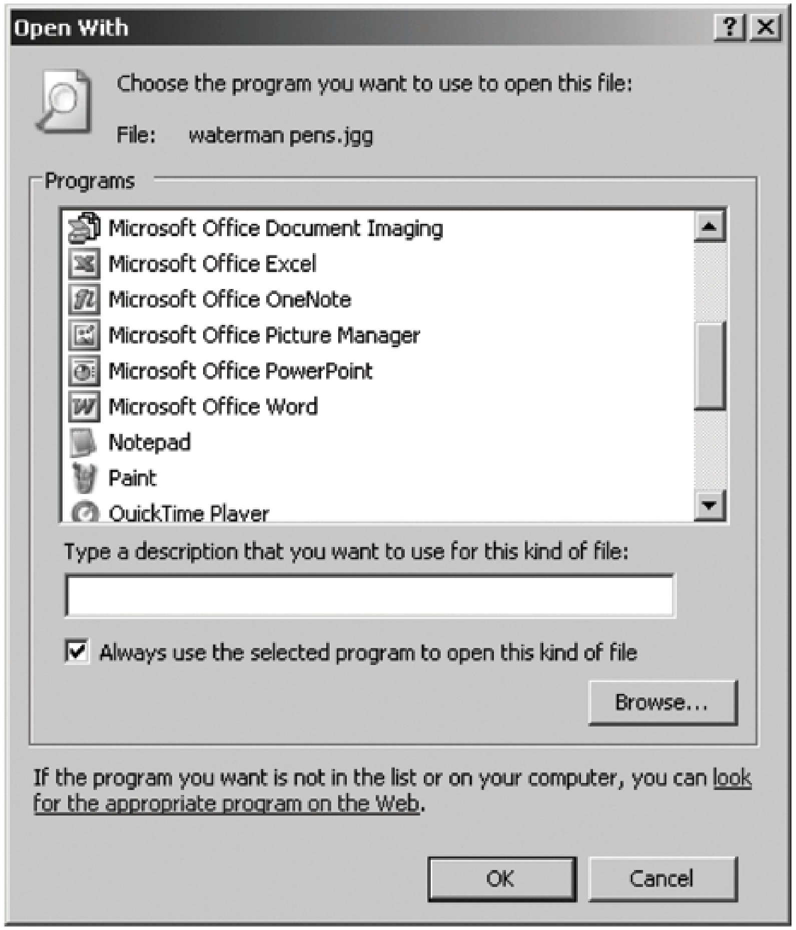
File-Naming Conventions (continued)

- **Extensions**
  - Appended to relative filename
    - Two to three characters
    - Separated by period
    - Identifies file type or contents
  - Example
    - BASIA_TUNE.MPG
  - Unknown extension
    - Requires user intervention

File-Naming Conventions (continued)

(figure 8.5)

To open a file with an unrecognized extension, Windows asks the user to choose an application to associate with that type of file.



Open With

Choose the program you want to use to open this file:

File: waterman pens.jgg

Programs

- Microsoft Office Document Imaging
- Microsoft Office Excel
- Microsoft Office OneNote
- Microsoft Office Picture Manager
- Microsoft Office PowerPoint
- Microsoft Office Word
- Notepad
- Paint
- QuickTime Player

Type a description that you want to use for this kind of file:

[✓] Always use the selected program to open this kind of file

If the program you want is not in the list or on your computer, you can look for the appropriate program on the Web.

File-Naming Conventions (continued)

- Operating system specifics
  - Windows
    - Drive label and directory name, relative name, and extension
  - Network with Open/VMS Alpha
    - Node, volume or storage device, directory, subdirectory, relative name and extension, file version number
  - UNIX/Linux
    - Forward slash (root), first subdirectory, sub-subdirectory, file’s relative name

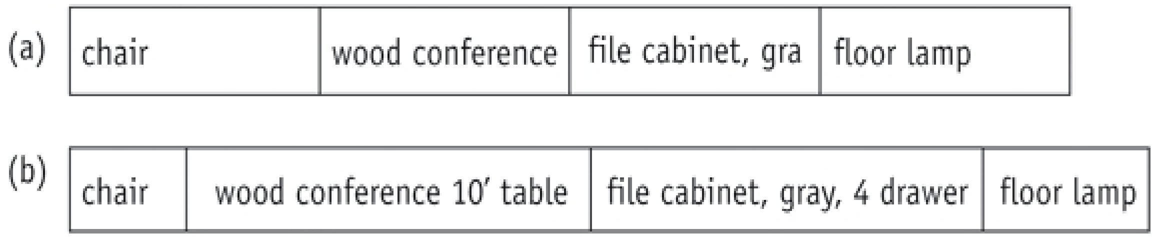
File Organization

- Arrangement of records within files
- All files composed of records
- Modify command
  - Request to access record within a file

Record Format

- **Fixed-length records**
  - Direct access: easy
  - Record size critical
  - Ideal for data files
- **Variable-length records**
  - Direct access: difficult
  - No empty storage space and no character truncation
  - File descriptor stores record format
  - Used with files accessed sequentially
    - Text files, program files
  - Used with files using index to access records

Record Format (continued)



(figure 8.6)

When data is stored in fixed-length fields (a), data that extends beyond the fixed size is truncated. When data is stored in a variable-length record format (b), the size expands to fit the contents, but it takes more time to access.

Physical File Organization

- Describes:
  - Record arrangement
  - Medium characteristics
- Magnetic disks file organization
  - Sequential, direct, indexed sequential
- File organization scheme selection considerations
  - Volatility of data
  - Activity of file
  - Size of file
  - Response time

Physical File Organization (continued)

- **Sequential record organization**
  - Records stored and retrieved serially
    - One after the other
  - Easiest to implement
  - File search: beginning until record found
  - Optimization features may be built into system
    - Select key field from record and sort before storage
    - Complicates maintenance algorithms
    - Preserve original order when records added, deleted

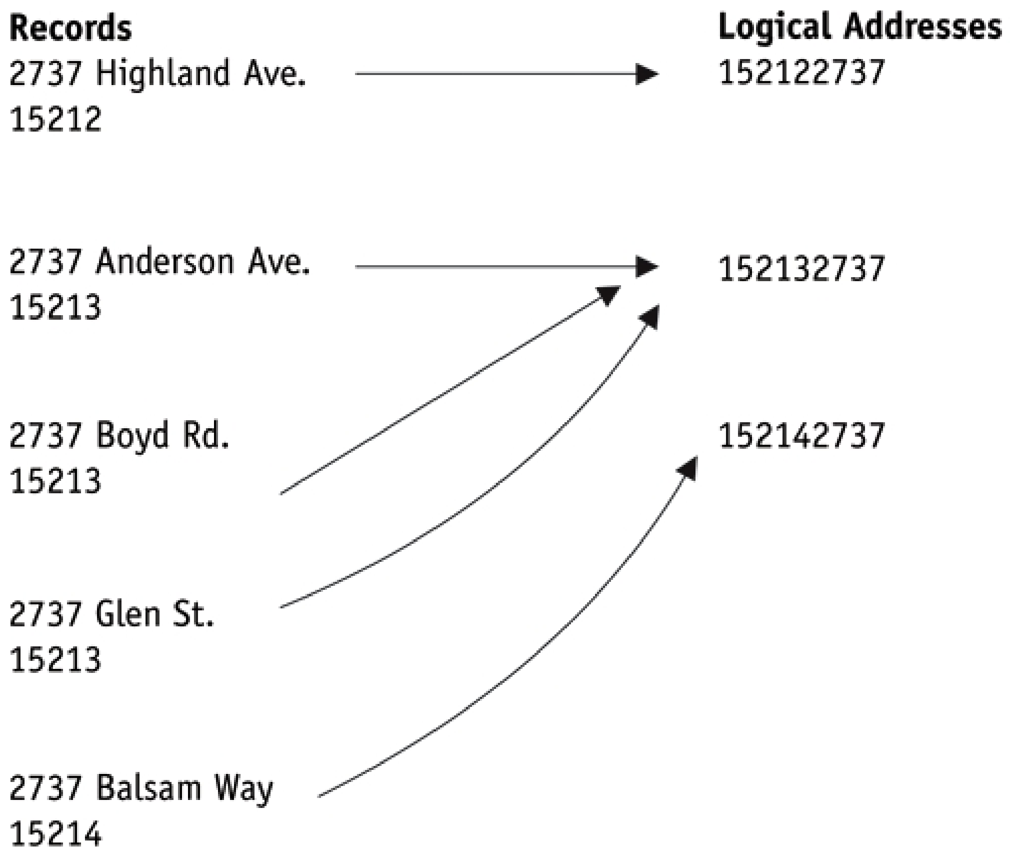
Physical File Organization (continued)

- **Direct record organization**
  - Uses direct access files
  - Direct access storage device implementation
    - Random organization
    - Random access files
  - **Relative address** record identification
    - Known as **logical addresses**
    - Computed when records stored, retrieved
  - Uses **hashing algorithms** to transform a **key field**

Physical File Organization (continued)

- Direct record organization (continued)
  - Advantages
    - Fast record access
    - Sequential access if starting at first relative address and incrementing to next record
    - Updated more quickly than sequential files
    - No preservation of records order
    - Adding, deleting records is quick
  - Disadvantages
    - Hashing algorithm collision
    - Similar keys

Physical File Organization (continued)



(figure 8.7)

The hashing algorithm causes a collision, when the algorithm, using a combination of street address and postal code, generates the same logical address (152132737) for three different records.

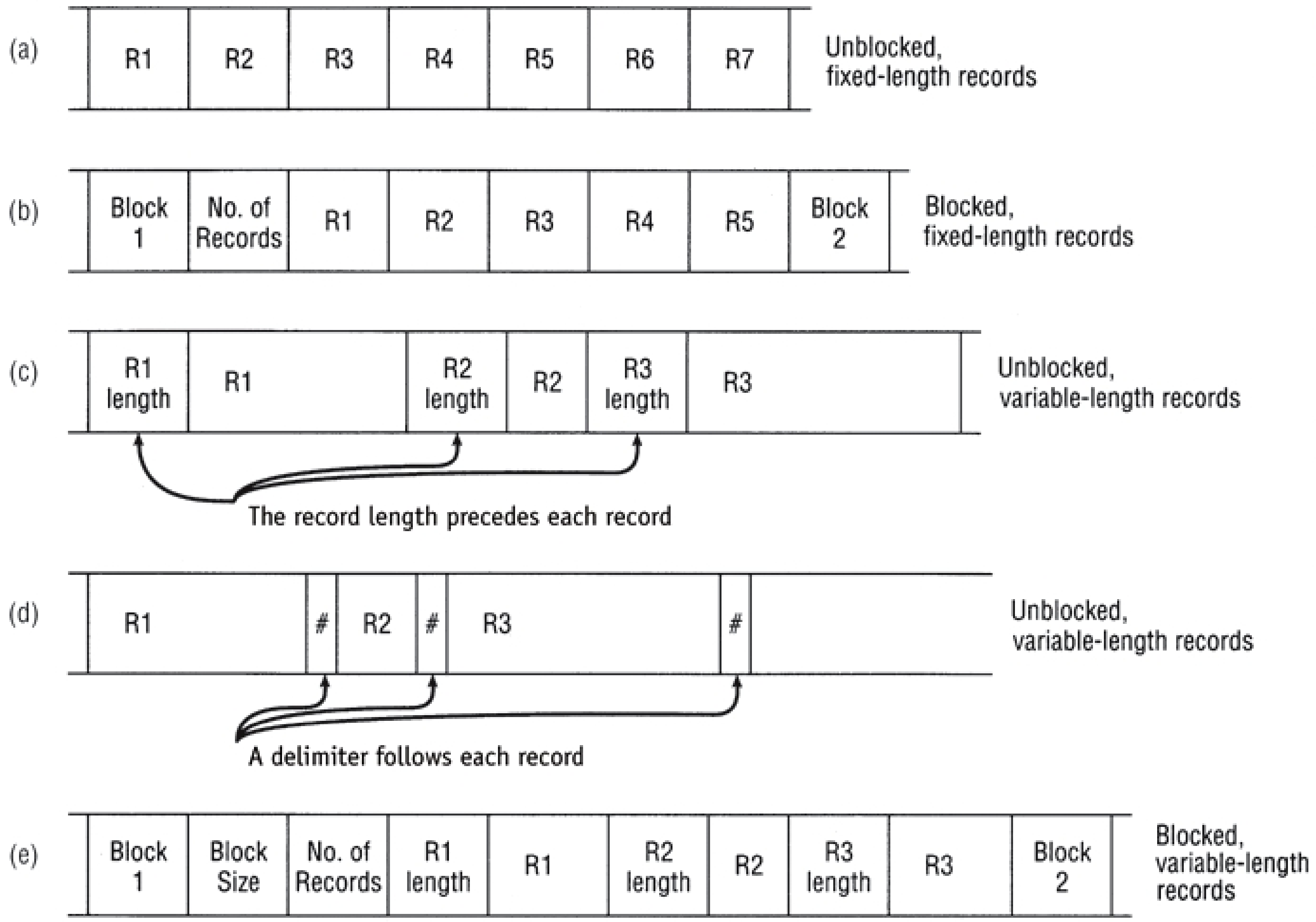
Physical File Organization (continued)

- **Indexed sequential record organization**
  - Best of sequential and direct access
  - ISAM software: creates, maintains
  - Advantage: no collisions (no hashing algorithm)
    - Generates index file for record retrieval
    - Divides ordered sequential file into equal sized blocks
    - Each entry in index file contains the highest record key and physical data block location
    - Search index file
  - Overflow areas

Physical Storage Allocation

- File manager works with files
  - As whole units
  - As logical units or records
- Within file
  - Records must have same format
  - Record length may vary
- Records subdivided into fields
- Application programs manage record structure
- File storage
  - Refers to record storage

Physical Storage Allocation (continued)

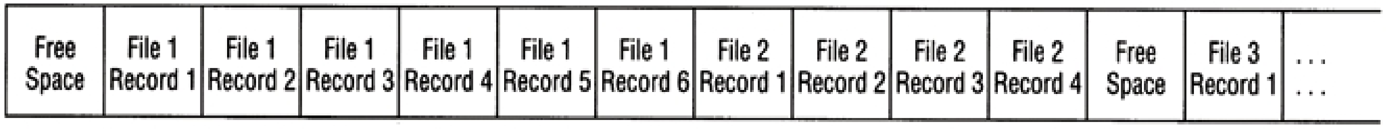

(a)
R1
R2
R3
R4
R5
R6
R7
Unblocked, fixed-length records
(b)
Block 1
No. of Records
R1
R2
R3
R4
R5
Block 2
Blocked, fixed-length records
(c)
R1 length
R1
R2 length
R2
R3 length
R3
Unblocked, variable-length records
The record length precedes each record
(d)
R1
#
R2
#
R3
#
Unblocked, variable-length records
A delimiter follows each record
(e)
Block 1
Block Size
No. of Records
R1 length
R1
R2 length
R2
R3 length
R3
Block 2
Blocked, variable-length records

(figure 8.8)

Every record in a file must have the same format but can be of different sizes, as shown in these five examples of the most common record formats. The supplementary information in (b), (c), (d), and (e) is provided by the File Manager, when the record is stored.

Contiguous Storage

- Records stored one after another
  - Advantages
    - Any record found once starting address, size known
    - Easy direct access
  - Disadvantages
    - Difficult file expansion, fragmentation



(figure 8.9)

Using this example of contiguous file storage, File 1 can't be expanded without being rewritten to a larger storage area. File 2 can be expanded, by only one record replacing the free space preceding File 3.

Noncontiguous Storage

- Files use any available disk storage space
- File records stored in contiguous manner
  - If enough empty space
- Remaining file records and additions
  - Stored in other disk sections (extents)
  - Extents
    - Linked together with pointers
    - Physical size determined by operating system
    - Usually 256 bytes

Noncontiguous Storage (continued)

- File extents linked in two ways
  - Storage level
    - Each extent points to next one in sequence
    - Directory entry
    - Filename, storage location of first extent, location of last extent, total number of extents (not counting first)
  - Directory level
    - Each extent listed with physical address, size, pointer to next extent
    - Null pointer indicates last one

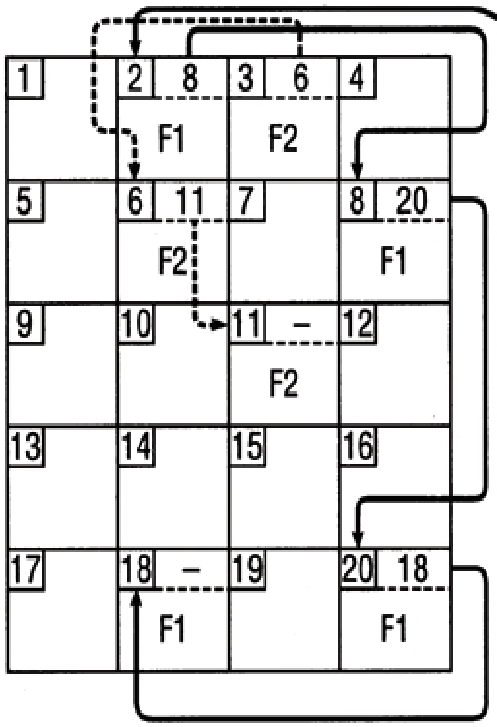
Noncontiguous Storage (continued)

- Advantage
  - Eliminates external storage fragmentation
  - Eliminates need for compaction
- Disadvantage
  - No direct access support
    - Cannot determine specific record’s exact location

Noncontiguous Storage (continued)

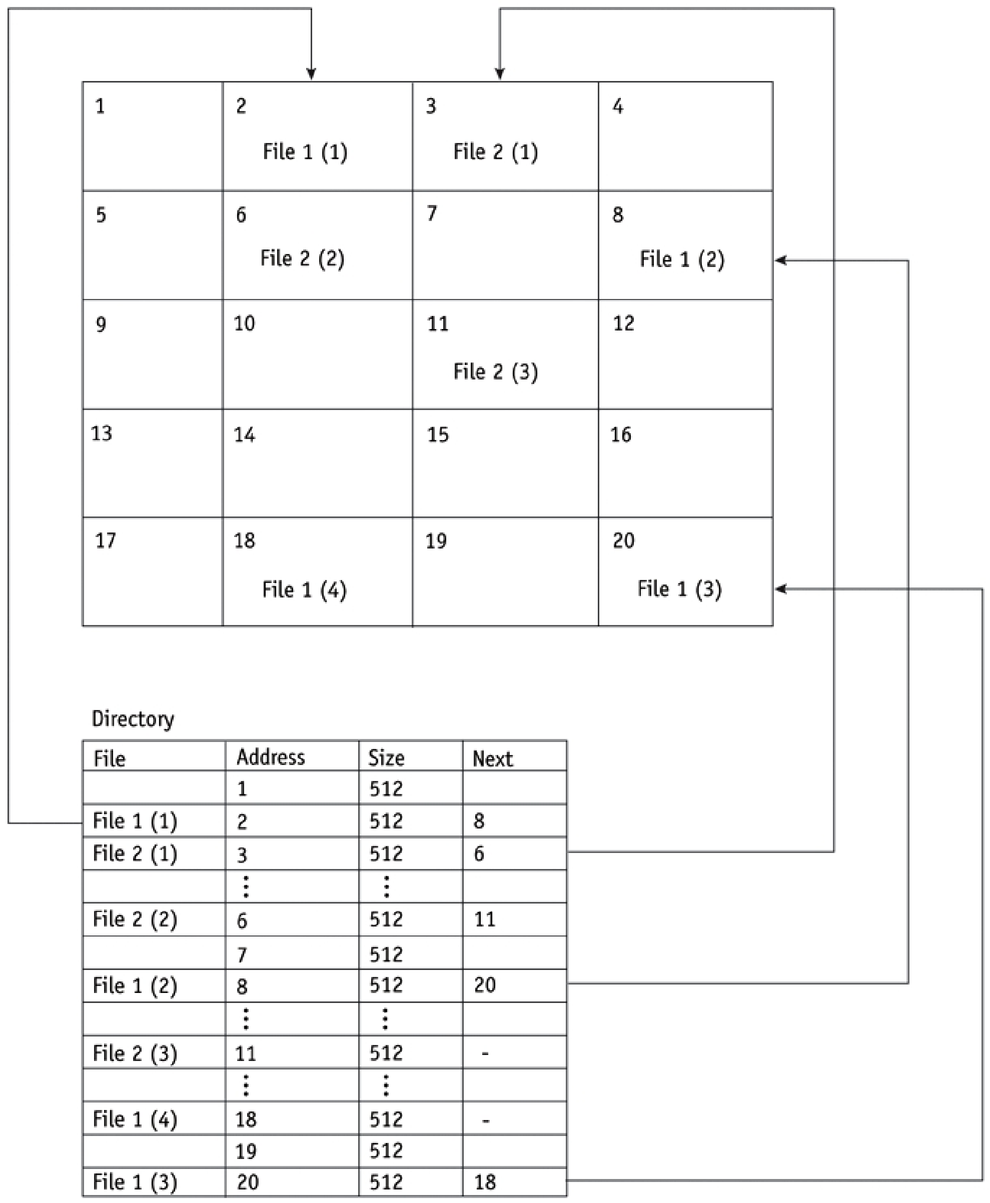
(figure 8.10)

Noncontiguous file storage with linking taking place at the storage level. File 1 starts in address 2 and continues in addresses 8, 20, and 18. The directory lists the file's starting address, ending address, and the number of extents it uses. Each block of storage includes its address and a pointer to the next block for the file, as well as the data itself.



Directory

File No.	Start	End	No. extents
1	2	18	3
2	3	11	2



File	Address	Size	Next
	1	512	
File 1 (1)	2	512	8
File 2 (1)	3	512	6
	⋮	⋮	
File 2 (2)	6	512	11
	7	512	
File 1 (2)	8	512	20
	⋮	⋮	
File 2 (3)	11	512	-
	⋮	⋮	
File 1 (4)	18	512	-
	19	512	
File 1 (3)	20	512	18

(figure 8.11)

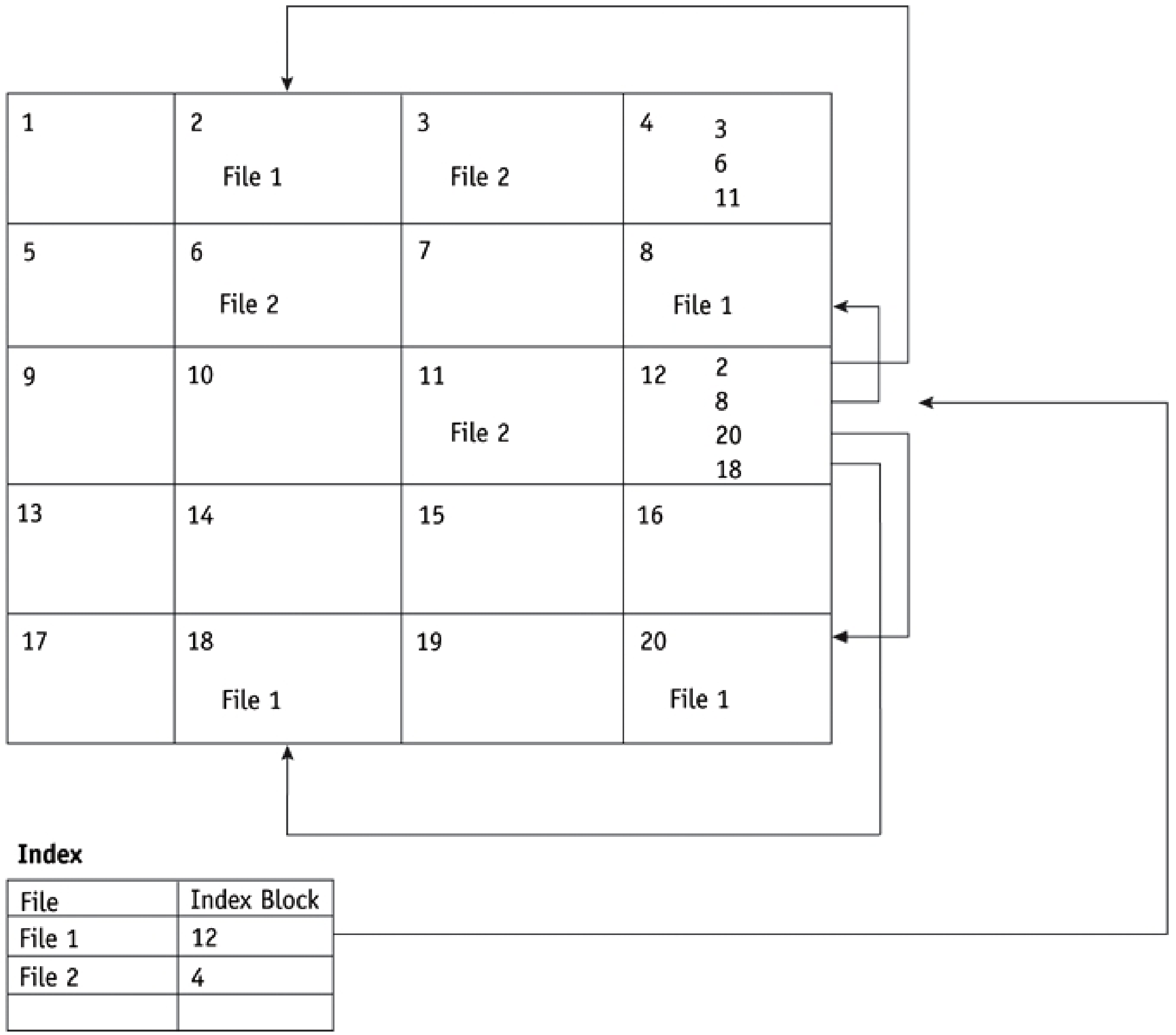
Noncontiguous storage allocation with linking taking place at the directory level for the files shown in Figure 8.10.

Indexed Storage

- Allows direct record access
  - Brings pointers together
    - Links every extent file into index block
- Every file has own index block
  - Disk sector addresses for file
  - Lists entry in order sectors linked
- Supports sequential and direct access
- Does not necessarily improve storage space use
- Larger files experience several index levels

(figure 8.12)

Indexed storage allocation with a one-level index, allowing direct access to each record for the files shown in Figures 8.10 and 8.11.



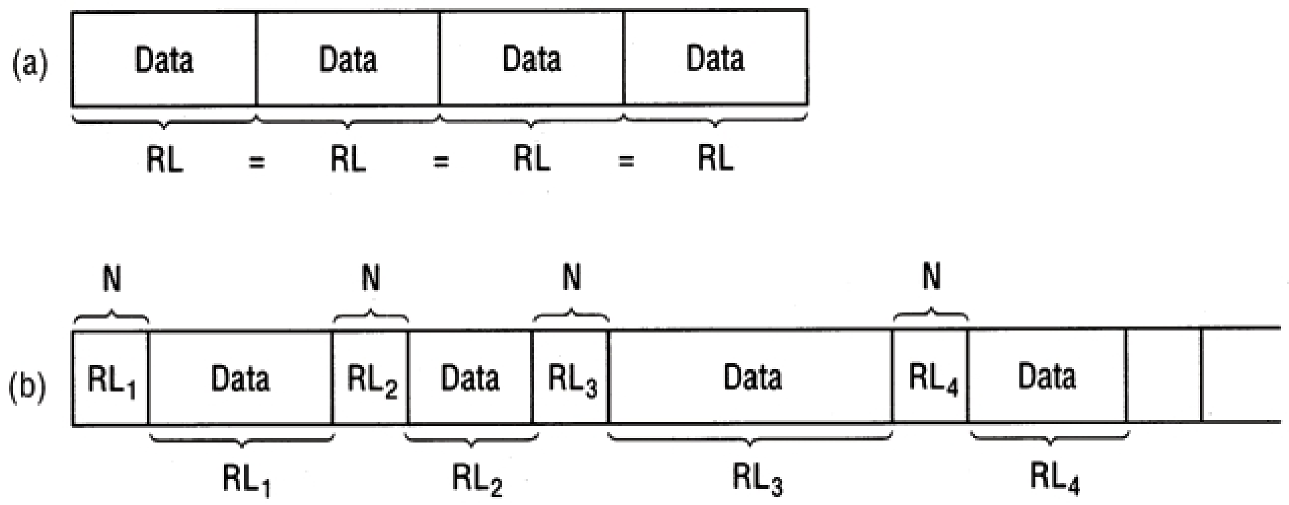
Index

File	Index Block
File 1	12
File 2	4

Access Methods

- Dictated by a file organization
- Most flexibility: indexed sequential files
- Least flexible: sequential files
- Sequential file organization
  - Supports only sequential access
    - Records: fixed or variable length
- Access next sequential record
  - Use address of last byte read
- **Current byte address (CBA)**
  - Updated every time record accessed

Access Methods (continued)



(figure 8.13)

Fixed- Versus variable-length records. (a) Fixed-length records have the same number of bytes, so record length (RL) is a constant. (b) With variable-length records RL_k isn't a constant. Therefore, it's recorded on the sequential media, such as magnetic tape, immediately preceding each record.

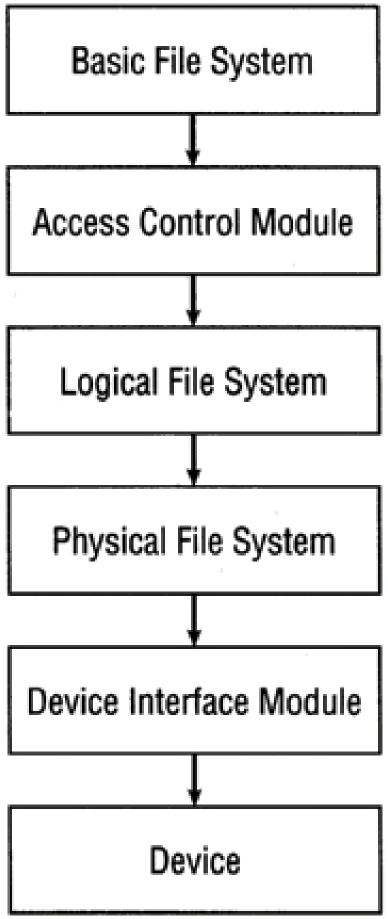
Sequential Access

- Update CBA
- Fixed-length records
  - Increment CB
    - CBA = CBA + RL
- Variable-length records
  - Add length of record (RL_K) plus numbers of bytes used to hold record to CBA
    - $CBA = CBA + N + RL_k$

Direct Access

- Fixed-length records (RN: desired record number)
  - CBA = (RN – 1) * RL
- Variable-length records
  - Virtually impossible
    - Address of desired record cannot be easily computed
  - Requires sequential search through records
  - Keep table of record numbers and CBAs
- Indexed sequential file
  - Accessed sequentially or directly
  - Index file searched for pointer to data block

Levels in a File Management System



(figure 8.14)

Typical modules of a file management system showing how information is passed from the File Manager at the top of the hierarchy to the Device Manager at the bottom.

Levels in a File Management System (continued)

- Level implementation
  - Structured and modular programming techniques
  - Hierarchical
    - Highest module passes information to lower module
- Modules further subdivided
  - More specific tasks
- Uses information of basic file system
  - Logical file system transforms record number to byte address

Levels in a File Management System (continued)

- **Verification** at every level
  - Directory level
    - File system checks if requested file exists
  - Access control verification module
    - Determines whether access allowed
  - Logical file system
    - Checks if requested byte address within file limits
  - Device interface module
    - Checks if storage device exists

Access Control Verification Module

- File sharing
  - Data files, user-owned program files, system files
  - Advantages
    - Save space, synchronized updates, resource efficiency
  - Disadvantage
    - Need to protect file integrity
  - Five possible file actions
    - READ only, WRITE only, EXECUTE only, DELETE only, combination
  - Four methods

Access Control Matrix

- Advantages
  - Easy to implement
  - Works well in system with few files, users

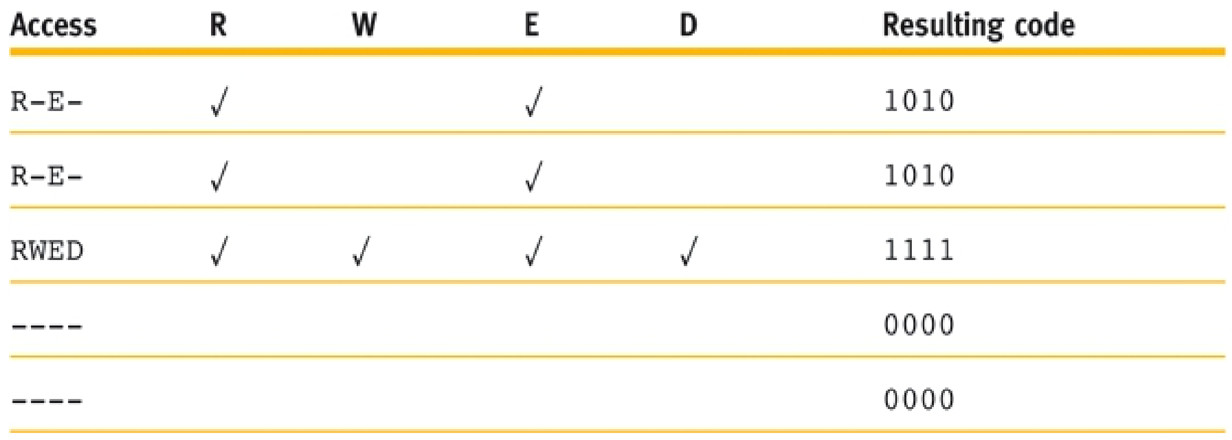
(table 8.2)

The access control matrix showing access rights for each user for each file. User 1 is allowed unlimited access to File 1 but is allowed only to read and execute File 4 and is denied access to the three other files. R = Read Access, W = Write Access, E = Execute Access, D = Delete Access, and a dash (-) = Access Not Allowed.

	User 1	User 2	User 3	User 4	User 5
File 1	RWED	R-E-	----	RWE-	--E-
File 2	----	R-E-	R-E-	--E-	----
File 3	----	RWED	----	--E-	----
File 4	R-E-	----	----	----	RWED
File 5	----	----	----	----	RWED

Access Control Matrix (continued)

- Disadvantages
  - As files and user increase, matrix increases
    - Possibly beyond main memory capacity
  - Wasted space: due to null entries



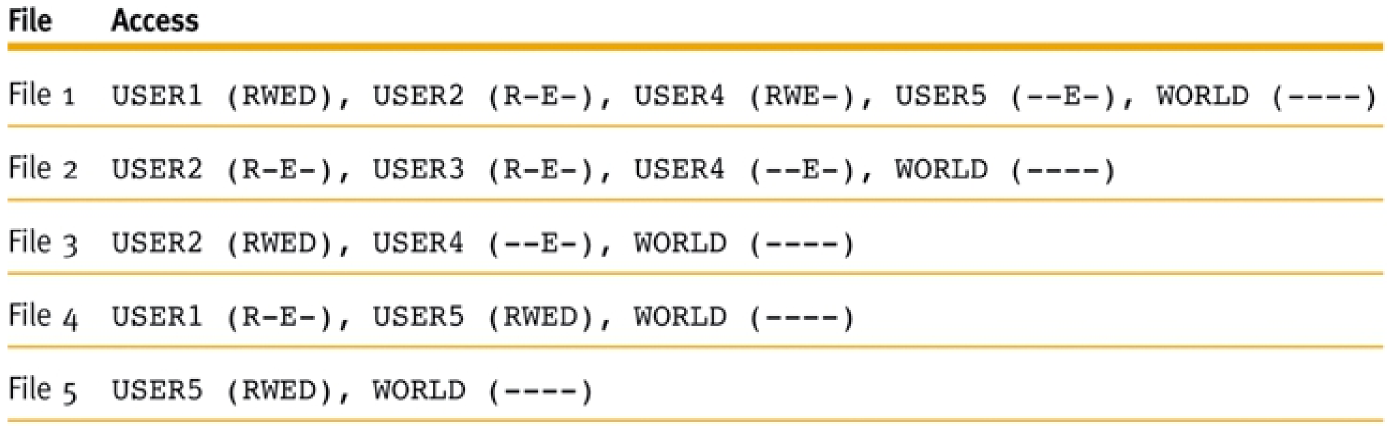
Access	R	W	E	D	Resulting code
R-E-	✓		✓		1010
R-E-	✓		✓		1010
RWED	✓	✓	✓	✓	1111
----					0000
----					0000

(table 8.3)

The five access codes for User 2 from Table 8.2 as represented in bits.

Access Control Lists

- Modification of access control matrix technique



File	Access
File 1	`USER1 (RWED), USER2 (R-E-), USER4 (RWE-), USER5 (--E-), WORLD (----)`
File 2	`USER2 (R-E-), USER3 (R-E-), USER4 (--E-), WORLD (----)`
File 3	`USER2 (RWED), USER4 (--E-), WORLD (----)`
File 4	`USER1 (R-E-), USER5 (RWED), WORLD (----)`
File 5	`USER5 (RWED), WORLD (----)`

(table 8.4)

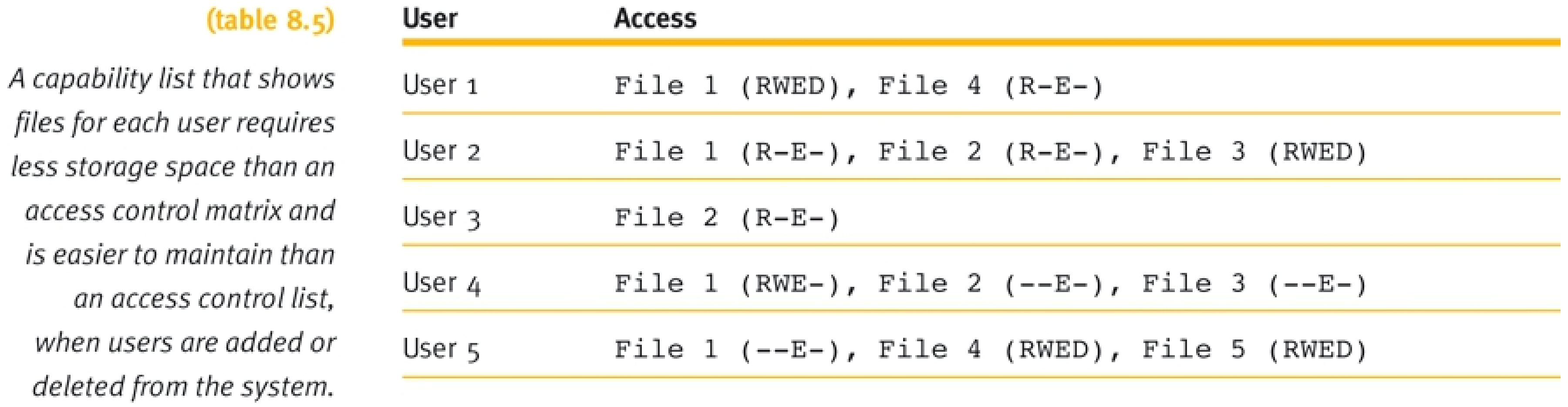
An access control list showing which users are allowed to access each file. This method requires less storage space than an access control matrix does and explicitly names each user allowed access to each file. WORLD indicates that access is allowed (or not allowed) to all users.

Access Control Lists (continued)

- Contains user names granted file access
  - User denied access grouped under “WORLD”
- Shorten list by categorizing users
  - SYSTEM
    - Personnel with unlimited access to all files
  - OWNER
    - Absolute control over all files created in own account
  - GROUP
    - All users belonging to appropriate group have access
  - WORLD
    - All other users in system

Capability Lists

- Lists every user and files each has access to
- Can control access to devices as well as to files
- Most common



(table 8.5)

A capability list that shows files for each user requires less storage space than an access control matrix and is easier to maintain than an access control list, when users are added or deleted from the system.

User	Access
User 1	File 1 (RWED), File 4 (R-E-)
User 2	File 1 (R-E-), File 2 (R-E-), File 3 (RWED)
User 3	File 2 (R-E-)
User 4	File 1 (RWE-), File 2 (--E-), File 3 (--E-)
User 5	File 1 (--E-), File 4 (RWED), File 5 (RWED)

Lockwords

- **Lockword**
  - Similar to password
  - Protects single file
- Advantage
  - Requires smallest storage amount for file protection
- Disadvantages
  - Guessable, passed on to unauthorized users
  - Does not control type of access
    - Anyone who knows lockword can read, write, execute, delete file

Data Compression

- A technique used to save space in files
- Text decompression
- Other decompression schemes

Text Compression

- **Records with repeated characters**
  - Repeated characters are replaced with a code
- **Repeated terms**
  - Compressed using symbols to represent most commonly used words
  - University student database common words
    - Student, course, grade, department each be represented with single character
- **Front-end compression**
  - Entry takes given number of characters from previous entry that they have in common

Other Compression Schemes

- Large files
  - Video and music
    - ISO MPEG standards
  - Photographs
- ISO
  - International Organization for Standardization

Summary

- File manager
  - Controls every file and processes user commands
  - Manages access control procedures
    - Maintain file integrity and security
  - File organizations
    - Sequential, direct, indexed sequential
  - Physical storage allocation schemes
    - Contiguous, noncontiguous, indexed
  - Record types
    - Fixed-length versus variable-length records
  - Four access methods