Chapter 8

File Management

Understanding Operating Systems, Fourth Edition

Objectives

You will be able to describe:

- 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
- Comparisons of sequential and direct file access

Objectives (continued)

You will be able to describe:

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

File Management

- File Manager controls every file in system
- **Efficiency of File Manager depends on:**
  - How system’s files are organized (sequential, direct, or indexed sequential)
  - How they’re stored (contiguously, noncontiguously, or indexed)
  - How each file’s records are structured (fixed-length or variable-length)
  - How access to these files is controlled

The File Manager

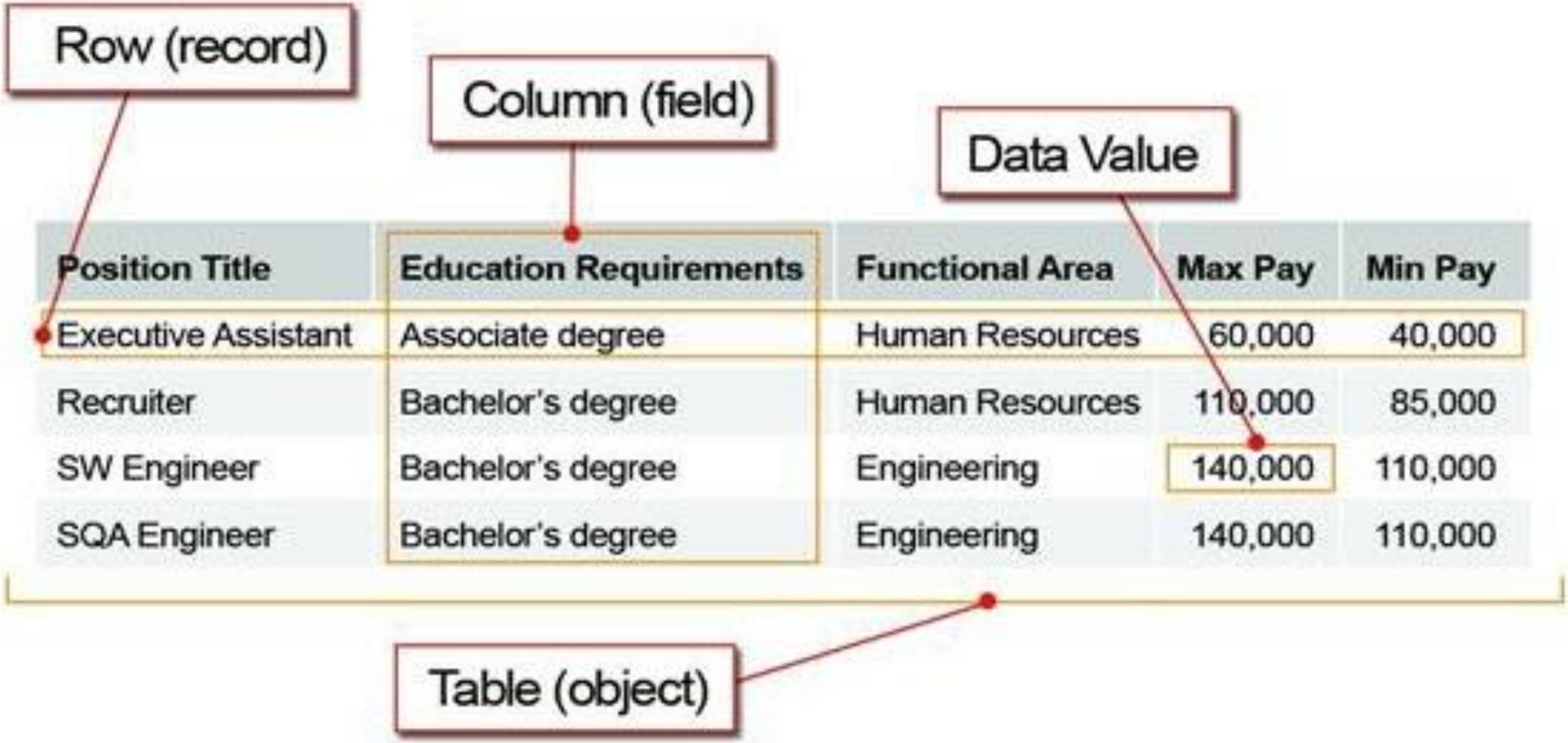
- **File Manager** is the software responsible for creating, deleting, modifying, and controlling access to files
  - Manages the resources used by files
- **Responsibilities of File Managers:**
  - Keep track of where each file is stored
  - Use a policy to determine where and how files will be stored
    - Efficiently use available storage space
    - Provide efficient access to files

The File Manager (continued)

- **Responsibilities of File Managers:** (continued)
  - Allocate each file when a user has been cleared for access to it, then record its use
  - Deallocate file when it is returned to storage and communicate its availability to others waiting for it

The File Manager (continued)

- **Definitions:**
  - **Field:** Group of related bytes that can be identified by user with name, type, and size
  - **Record:** Group of related fields
  - **File:** Group of related records that contains information used by specific application programs to generate reports
    - Sometimes called flat file; has no connections to other files
  - **Database:** Groups of related files that are interconnected at various levels to give users flexibility of access to the data stored



Position Title	Education Requirements	Functional Area	Max Pay	Min Pay
Executive Assistant	Associate degree	Human Resources	60,000	40,000
Recruiter	Bachelor's degree	Human Resources	110,000	85,000
SW Engineer	Bachelor's degree	Engineering	140,000	110,000
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000

The File Manager (continued)

- **Program files:** Contain instructions
- **Data files:** Contain data
- **Directories:** Listings of filenames and their attributes
- Every program and data file accessed by computer system, and every piece of computer software, is treated as a file
- File Manager treats all files exactly the same way as far as storage is concerned

Interacting with the File Manager

- User communicates with File Manager via specific commands that may be:
  - **Embedded in the user’s program**
    - OPEN, CLOSE, READ, WRITE, and MODIFY
  - **Submitted interactively by the user**
    - CREATE, DELETE, RENAME, and COPY
- Commands are **device independent**
  - User doesn’t need to know its exact physical location on disk pack or storage medium to access a file

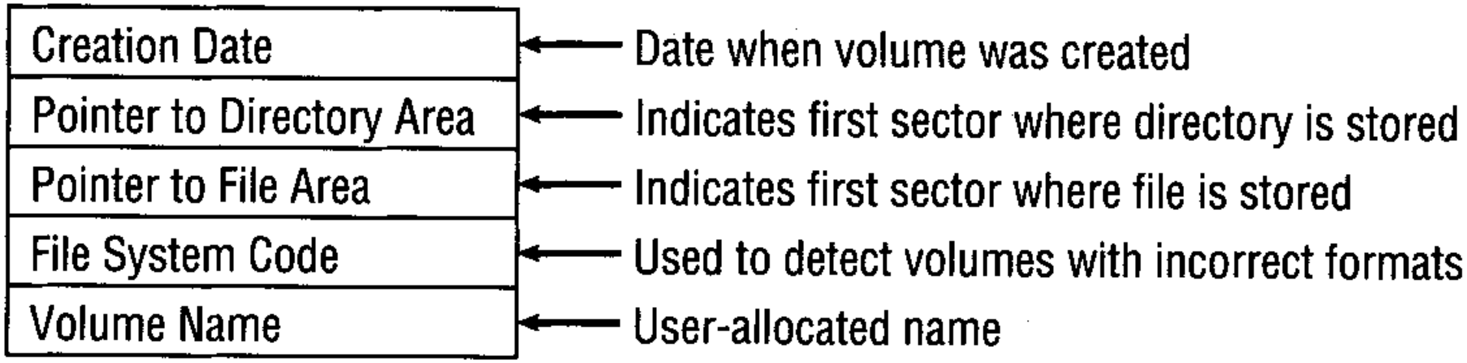
Interacting with the File Manager (continued)

- Each logical command is broken down into sequence of low-level signals that
  - Trigger step-by-step actions performed by device
  - Supervise progress of operation by testing status
- Users don't need to include in each program the low-level instructions for every device to be used
- Users can manipulate their files by using a simple set of commands (e.g., OPEN, CLOSE, READ, WRITE, and MODIFY)

Typical Volume Configuration

- **Volume:** Each secondary storage unit (removable or non-removable)
  - Each volume can contain many files called **multifile volumes**
  - Extremely large files are contained in many volumes called **multivolume files**
- Each volume in a system is given a **name**
  - File Manager writes name & other descriptive info on an easy-to-access place on each unit

Typical Volume Configuration (continued)



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

Figure 8.1: Volume descriptor, stored at the beginning of each volume

Typical Volume Configuration (continued)

- **Master file directory (MFD):** Stored immediately after volume descriptor and lists:
  - Names and characteristics of every file in volume
    - File names can refer to program files, data files, and/or system files
  - Subdirectories, if supported by File Manager
  - Remainder of the volume used for file storage

Typical Volume Configuration (continued)

- Disadvantages of a **single directory per volume** as supported by early operating systems:
  - Long time to search for an individual file
  - Directory space would fill up before the disk storage space filled up
  - Users couldn’t create subdirectories
  - Users couldn’t safeguard their files from other users
  - Each program in the directory needed a unique name, even those directories serving many users

About Subdirectories

Subdirectories:

- **Semi-sophisticated File Managers** create MFD for each volume with entries for files and subdirectories
- Subdirectory created when user opens account to access computer
- Improvement from single directory scheme
- Still can’t group files in a logical order to improve accessibility and efficiency of system

About Subdirectories (continued)

Subdirectories:

- **Today’s File Managers** allow users to create subdirectories (**Folders**)
  - Allows related files to be grouped together
- Implemented as an **upside-down tree**
  - Allows system to efficiently search individual directories
- Path to the requested file may lead through several directories

About Subdirectories (continued)

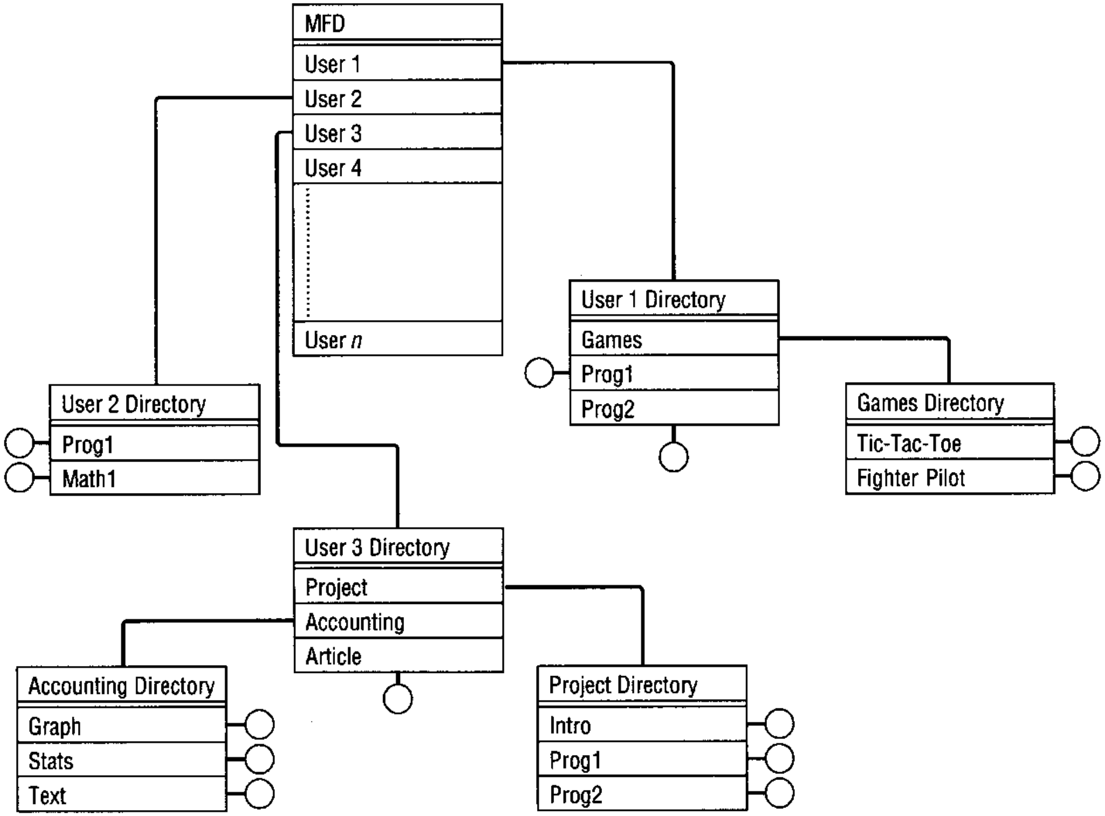


Figure 8.2: File directory tree structure

About Subdirectories (continued)

- **File descriptor** includes the following information:
  - Filename
  - File type
  - File size
  - File location
  - Date and time of creation
  - Owner
  - Protection information
  - Record size

File Naming Conventions

- **Absolute filename** (complete filename): Long name that includes all path info
- **Relative filename**: Short name seen in directory listings and selected by user when file is created
- Length of relative name and types of characters allowed is OS dependent
- **Extension:** Identifies type of file or its contents
  - e.g., BAT, COB, EXE, TXT, DOC
- Components required for a file’s complete name depend on the operating system

File Organization

- All files composed of records that are of two types:
  - **Fixed-length records:** Easiest to access directly
    - Ideal for data files
    - Record size critical
  - **Variable-length records**: Difficult to access directly
    - Don’t leave empty storage space and don’t truncate any characters
    - Used in files accessed sequentially (e.g., text files, program files) or files using index to access records
    - File descriptor stores record format

File Organization (continued)

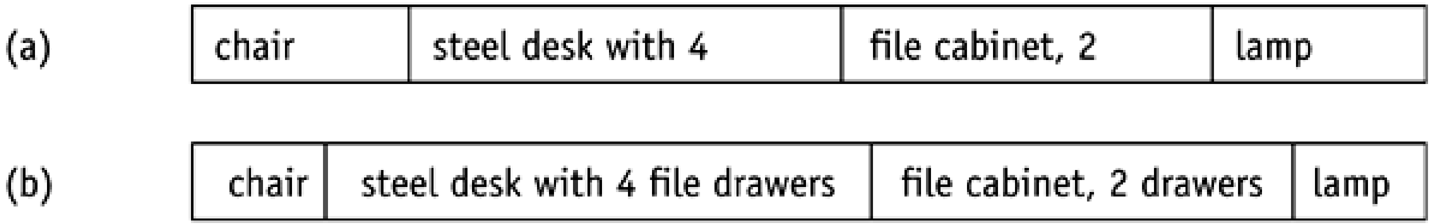
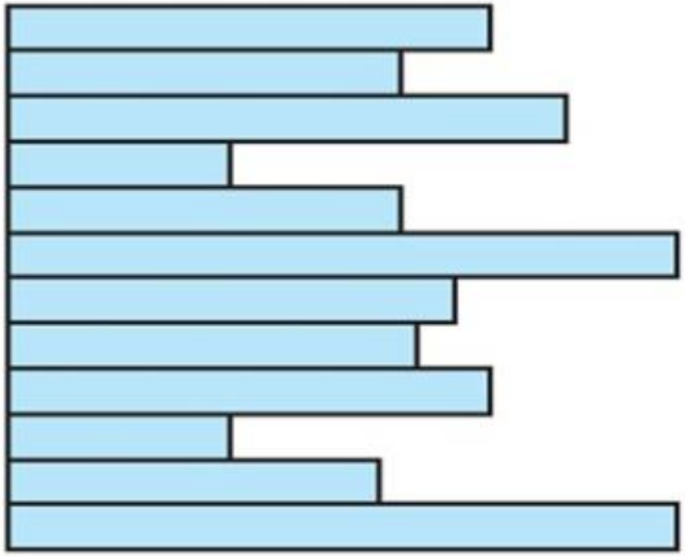
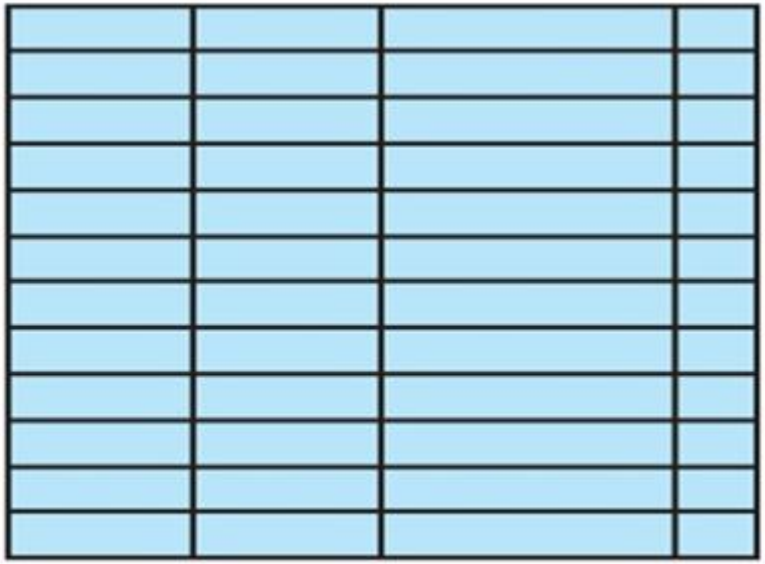


Figure 8.4: When data is stored in fixed-length fields (a), data that extends beyond the fixed size is truncated. When data is stored in a variable length record format (b), the size expands to fit the contents, but it takes more time to access.



Variable-length records
Variable set of fields



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

Physical File Organization

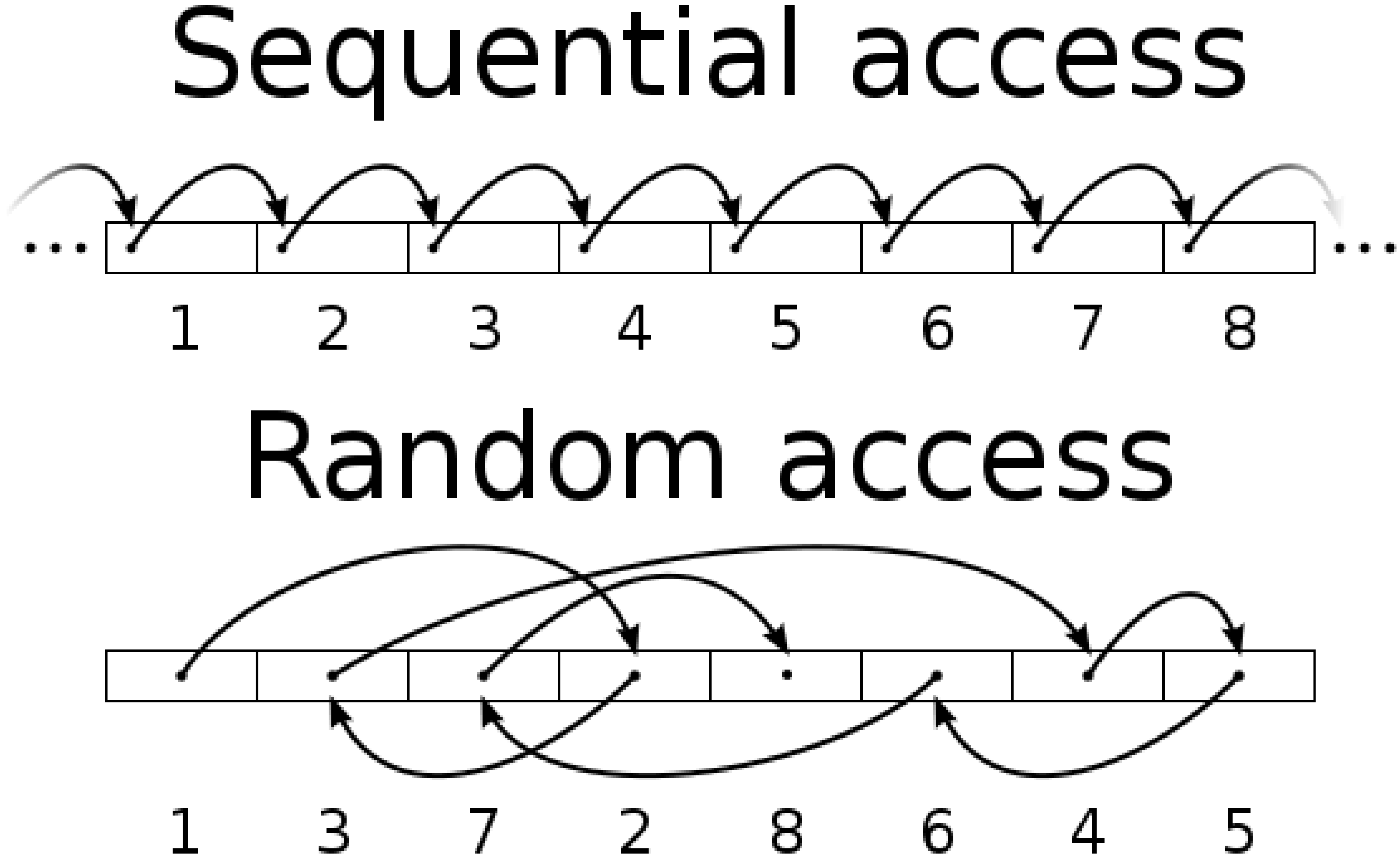
- The way records are arranged and the characteristics of the medium used to store them
- On magnetic disks, files can be organized as: sequential, direct, or indexed sequential
- Considerations in selecting a file organization scheme:
  - Volatility of the data
  - Activity of the file
  - Size of the file
  - Response time

Physical File Organization (continued)

- **Sequential record organization:** Records are stored and retrieved serially (one after the other)
  - Easiest to implement
  - File is searched from its beginning until the requested record is found
  - Optimization features may be built into system to speed search process
    - Select a key field from the record
  - Complicates maintenance algorithms
    - Original order must be preserved every time records are added or deleted

Physical File Organization (continued)

- **Direct record organization:** Uses direct access files; can be implemented only on direct access storage devices
  - Allows accessing of any record in any order without having to begin search from beginning of file
  - Records are identified by their **relative addresses** (addresses relative to beginning of file)
    - These **logical addresses** computed when records are stored and again when records are retrieved
  - Use hashing algorithms

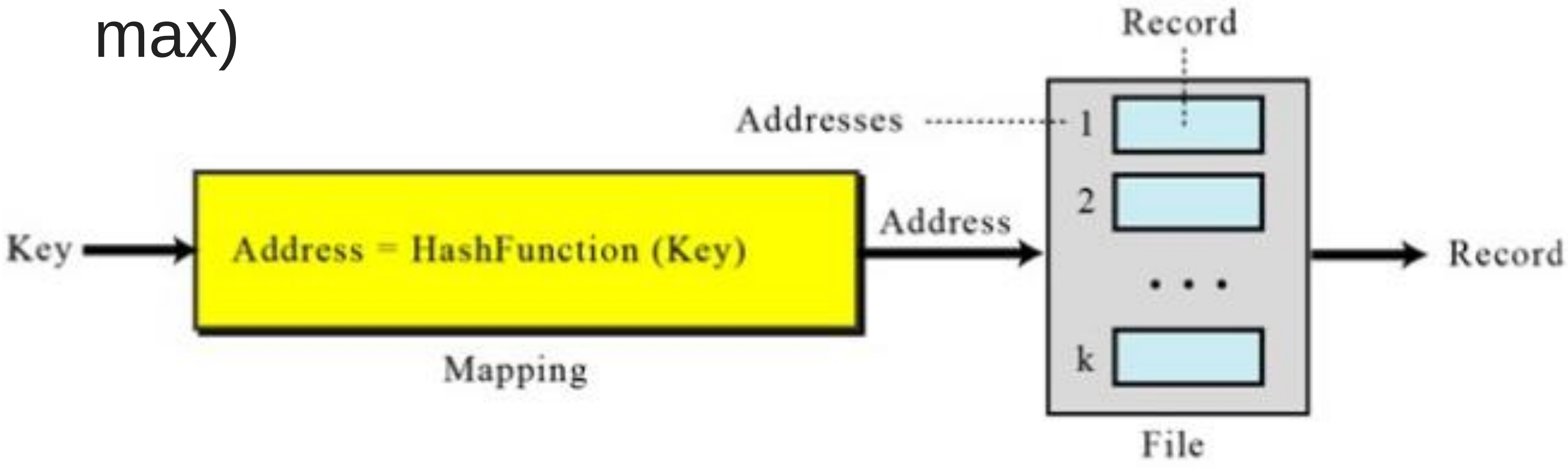

Sequential access
1
2
3
4
5
6
7
8
Random access
1
3
7
2
8
6
4
5

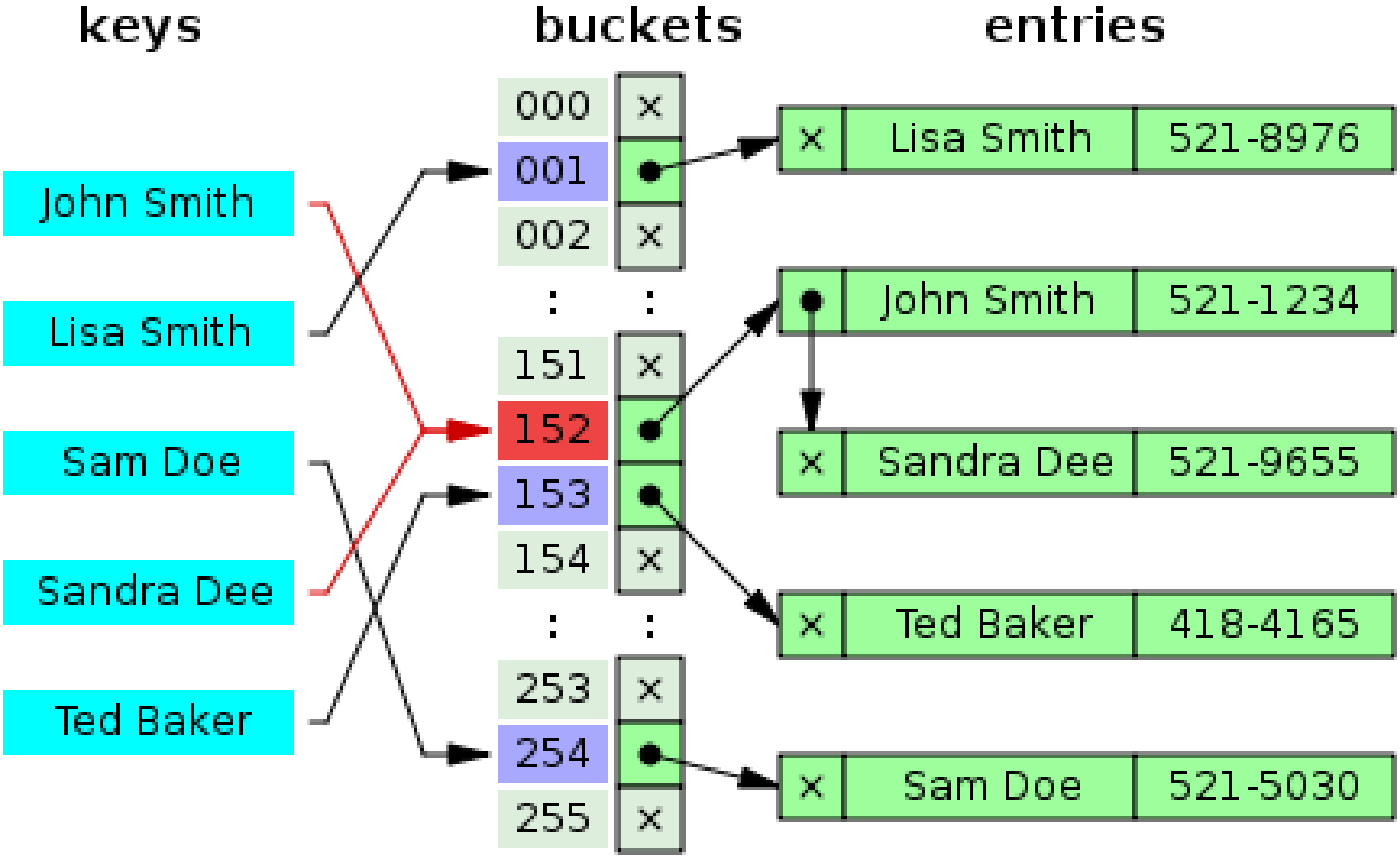
Physical File Organization (continued)

- **Advantages** of direct record organization:
  - Fast,
  - accessed sequentially
  - Quicker updated than sequential files
  - No need to preserve order of the records, so adding or deleting them takes very little time
- **Disadvantages** of direct record organization:
  - Collision in case of similar keys

Hashed Files

- Hashed files use a mathematical function to accomplish mapping of records.
- A file is hashed based on a unique field
- Each time a value is given to Hash function it returns the address of that record (a range 1 to max)



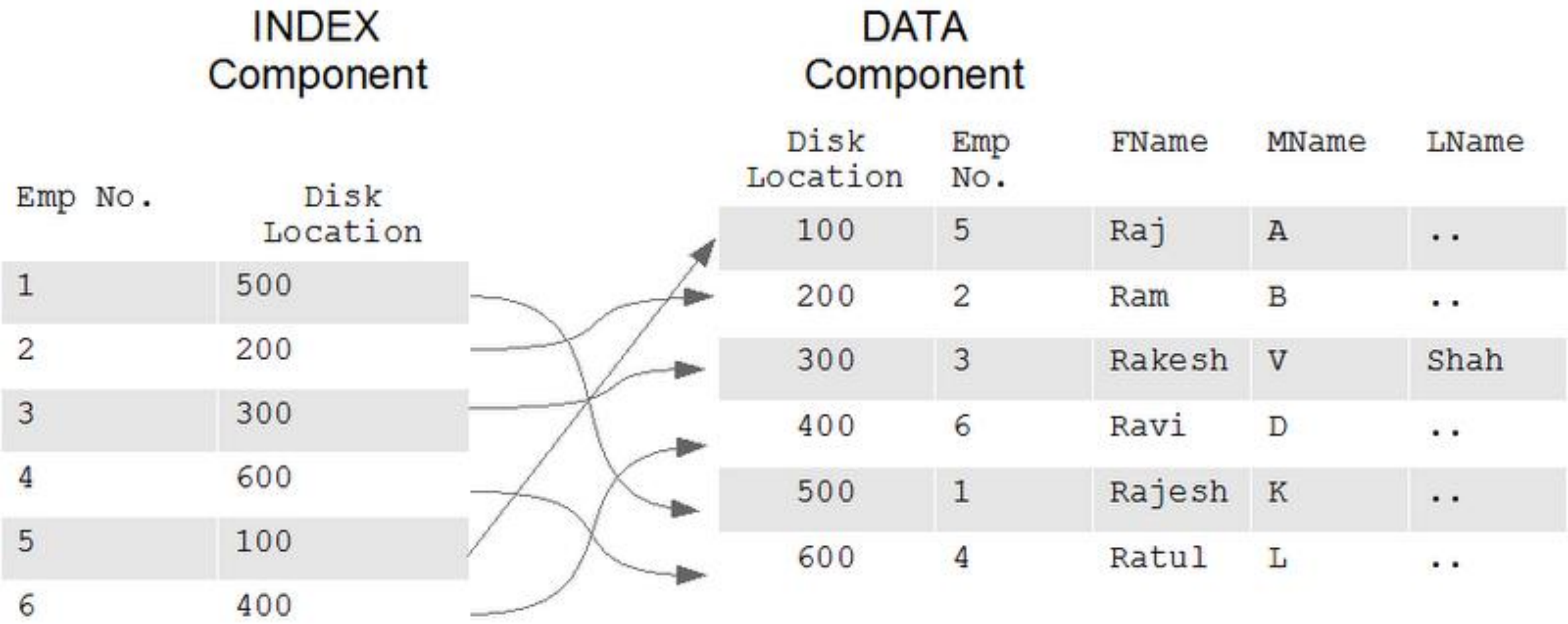


keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
:
:
151
152
153
154
:
:
253
254
255
x
Lisa Smith
521-8976
John Smith
521-1234
Sandra Dee
521-9655
Ted Baker
418-4165
Sam Doe
521-5030

Physical File Organization (continued)

- **Indexed sequential record organization:** generates index file for record retrieval
  - Combines best of sequential & direct access
  - Divides ordered sequential file into blocks of equal size
  - Each entry in index file contains highest record key and physical location of data block
  - Created and maintained through ISAM software (**Indexed Sequential Access Method**)
  - **Advantage:** Doesn’t create collisions

Indexed files

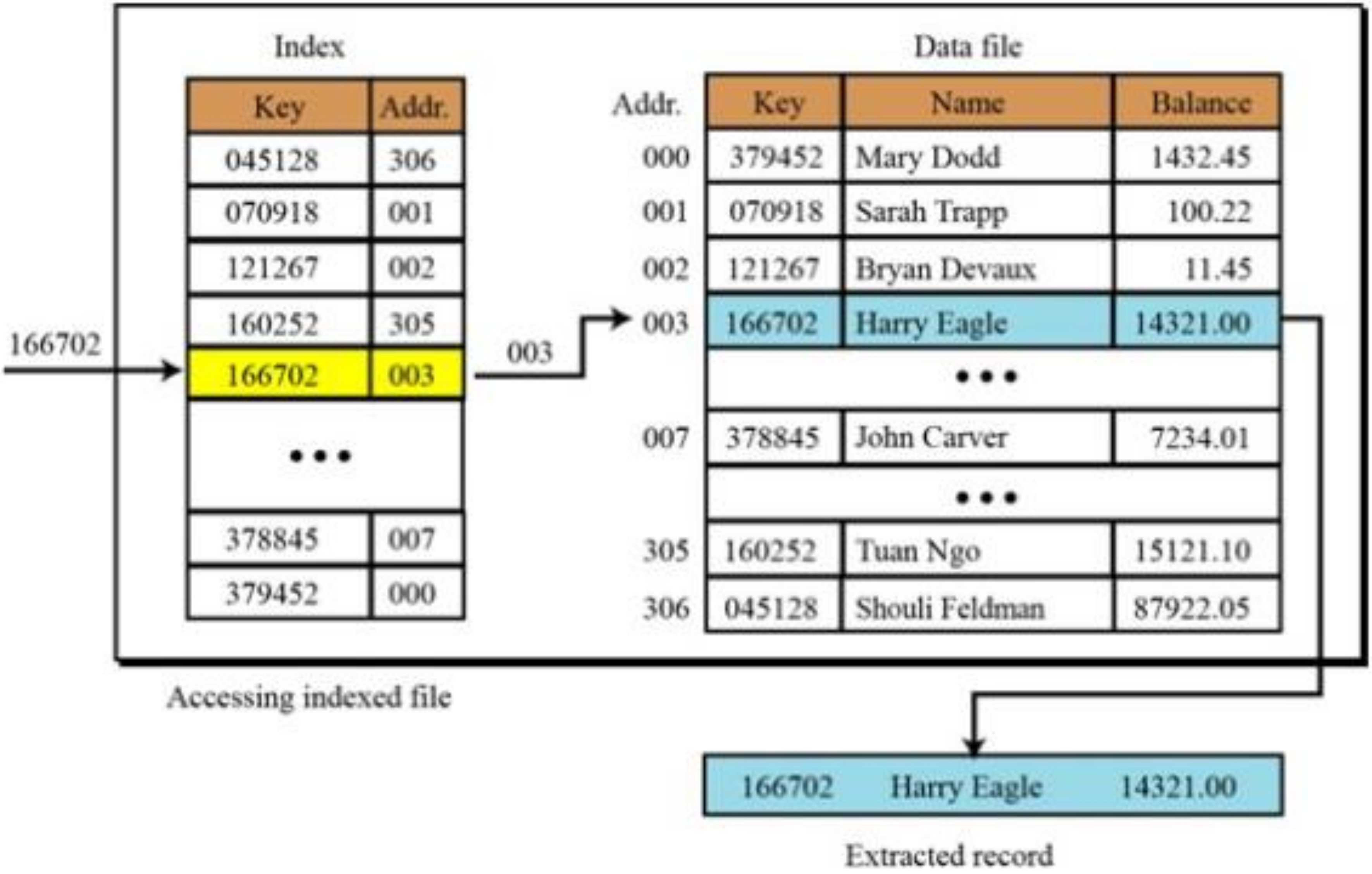


INDEX Component

Emp No.	Disk Location
1	500
2	200
3	300
4	600
5	100
6	400

DATA Component

Disk Location	Emp No.	FName	MName	LName
100	5	Raj	A	..
200	2	Ram	B	..
300	3	Rakesh	V	Shah
400	6	Ravi	D	..
500	1	Rajesh	K	..
600	4	Ratul	L	..



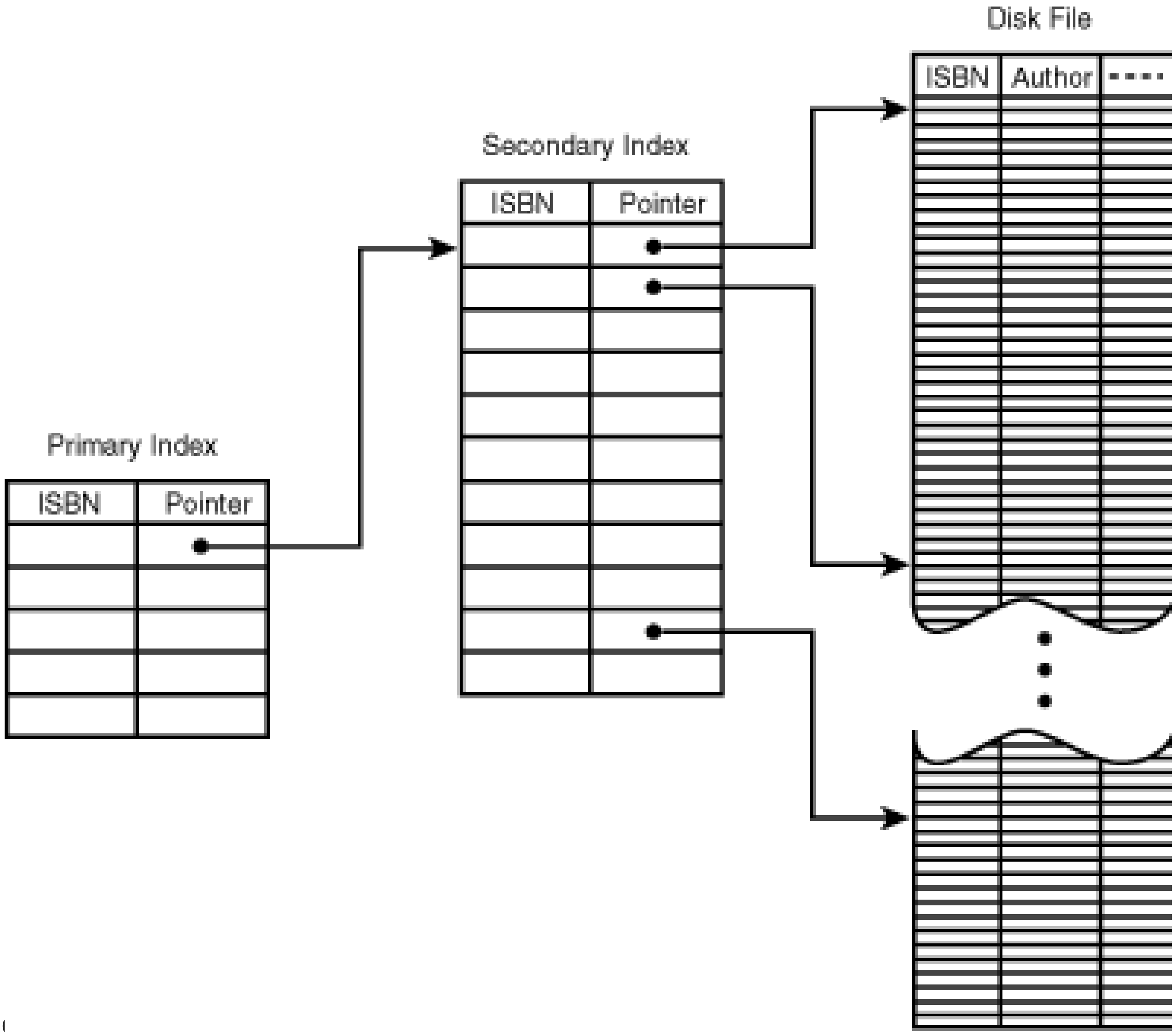
Index

Key	Addr.
045128	306
070918	001
121267	002
160252	305
166702	003
• • •	
378845	007
379452	000

Data file

Addr.	Key	Name	Balance
000	379452	Mary Dodd	1432.45
001	070918	Sarah Trapp	100.22
002	121267	Bryan Devaux	11.45
003	166702	Harry Eagle	14321.00
	• • •		
007	378845	John Carver	7234.01
	• • •		
305	160252	Tuan Ngo	15121.10
306	045128	Shouli Feldman	87922.05

166702	Harry Eagle	14321.00


Disk File
ISBN
Author

Secondary Index
ISBN
Pointer
Primary Index
ISBN
Pointer

Physical Storage Allocation

- File Manager must work with files not just as whole units but also as logical units or records
- Records within a file must have the same format but they can vary in length
- Records are subdivided into fields
- Record’s structure usually managed by application programs and not OS
- File storage actually refers to record storage

Physical Storage Allocation (continued)

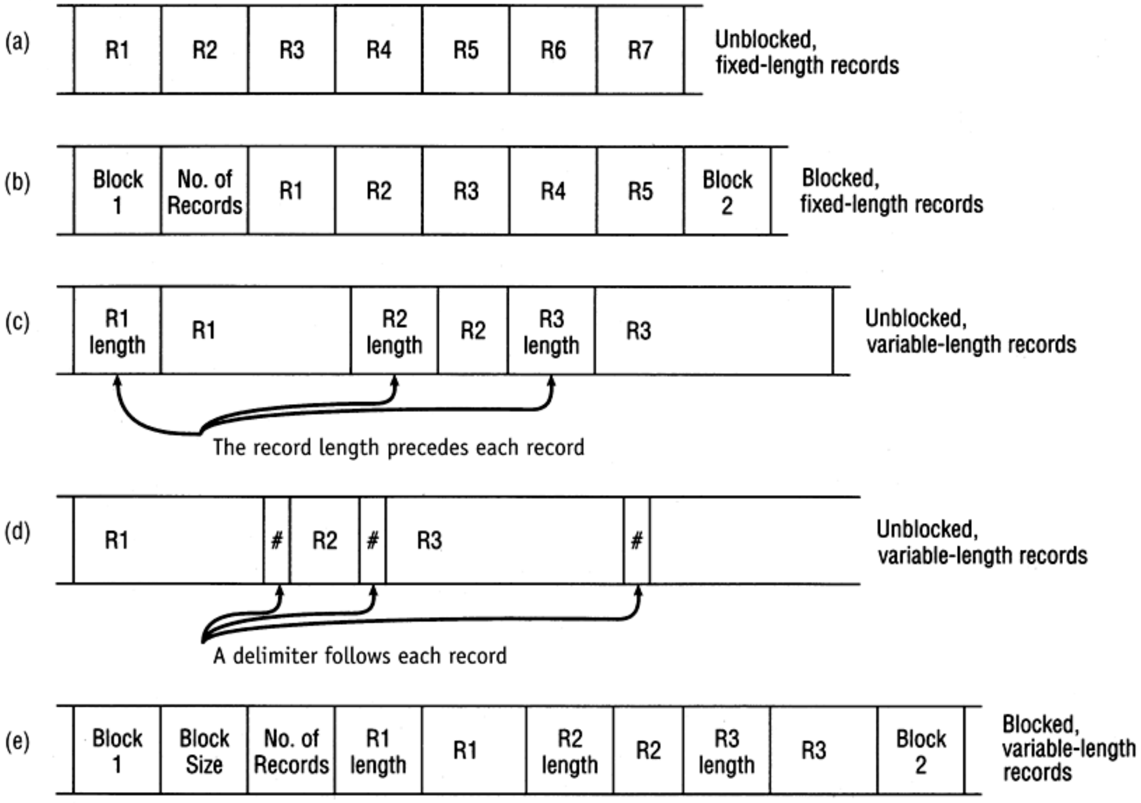
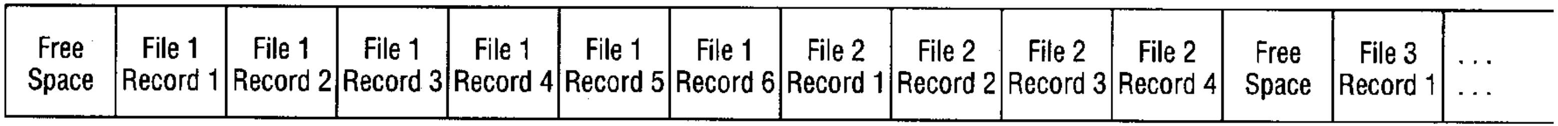

(a)
R1
R2
R3
R4
R5
R6
R7
Unblocked, fixed-length records
(b)
Block 1
No. of Records
R1
R2
R3
R4
R5
Block 2
Blocked, fixed-length records
(c)
R1 length
R1
R2 length
R2
R3 length
R3
Unblocked, variable-length records
The record length precedes each record
(d)
R1
#
R2
#
R3
#
Unblocked, variable-length records
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 8.6: Types of records in a file

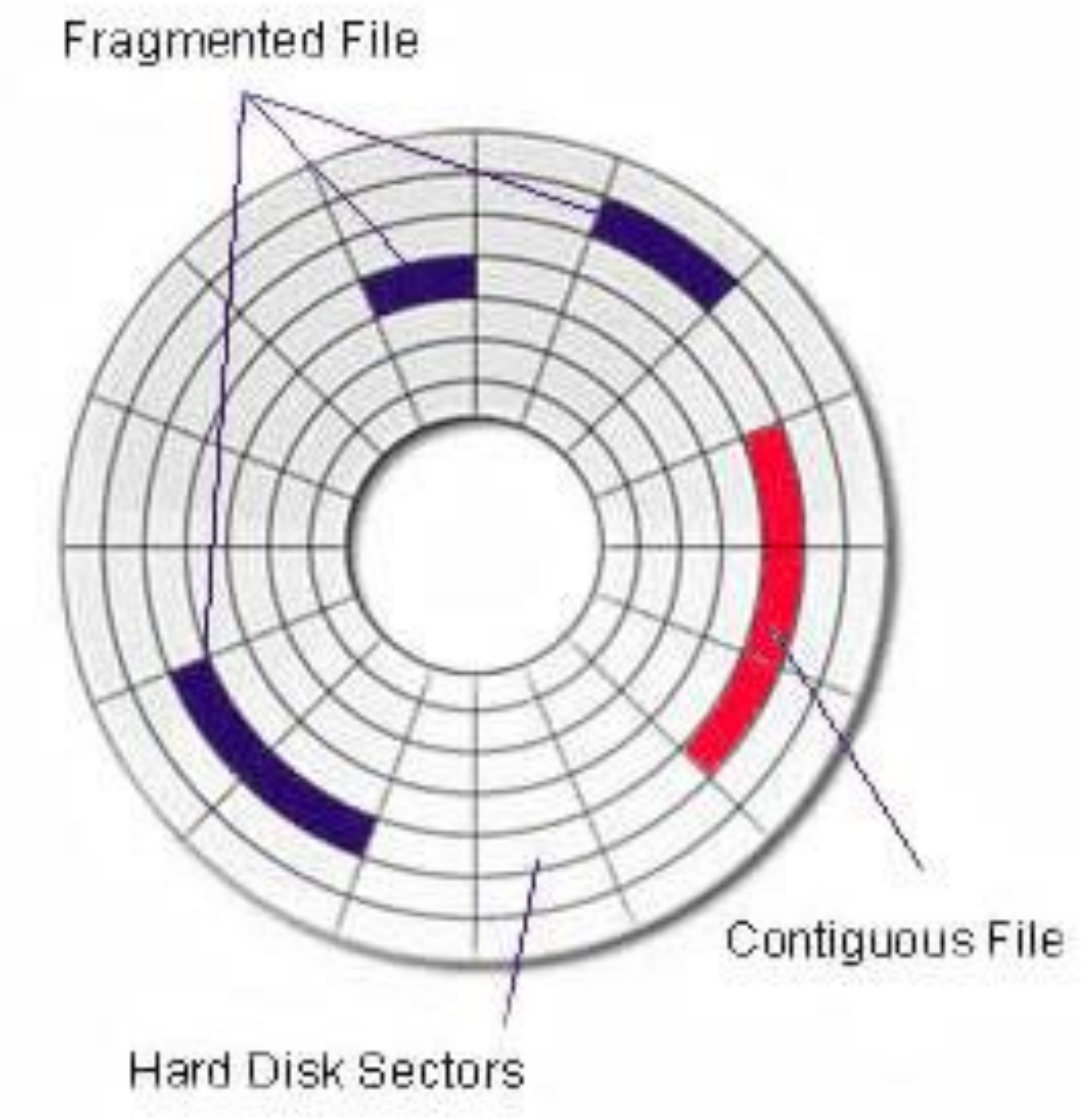
Contiguous Storage

- Records stored one after another
  - **Advantages:**
    - Any record can be found once starting address and size are known
    - Direct access easy as every part of file is stored in same compact area
  - **Disadvantages:**
    - Files can’t be expanded easily, and fragmentation



Free Space	File 1 Record 1	File 1 Record 2	File 1 Record 3	File 1 Record 4	File 1 Record 5	File 1 Record 6	File 2 Record 1	File 2 Record 2	File 2 Record 3	File 2 Record 4	Free Space	File 3 Record 1	. . . . . .

Figure 8.7: Contiguous storage


Fragmented File
Contiguous File
Hard Disk Sectors

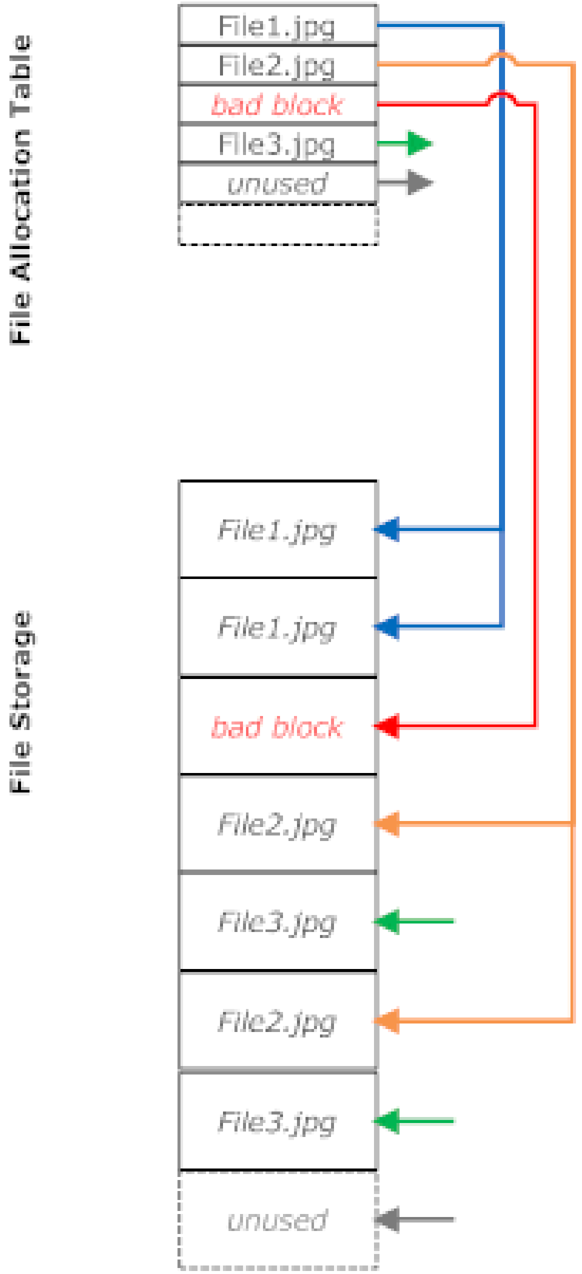
Noncontiguous Storage

- Allows files to use any available disk storage space
- File’s records are stored in a contiguous manner if enough empty space
- Any remaining records, and all other additions to file, are stored in other sections of disk (**extents**)
  - Linked together with pointers
  - Physical size of each extent is determined by OS (usually 256 bytes)

Noncontiguous Storage (continued)

- File extents are linked in following ways:
- **Linking at storage level:**
  - Each extent points to next one in sequence
  - Directory entry consists of filename, storage location of first extent, location of last extent, and total number of extents, not counting first
- **Linking at directory level:**
  - Each extent listed with its physical address, size, and pointer to next extent
  - A null pointer indicates that it's the last one

FAT Disk file structure (simplified)



The File Allocation Table (FAT) marks which data blocks are used by which file, which blocks are bad and which are unused.

A file takes up at least one data block but it can take up more.

If any part of the block has a fault then the entire block is marked as bad.

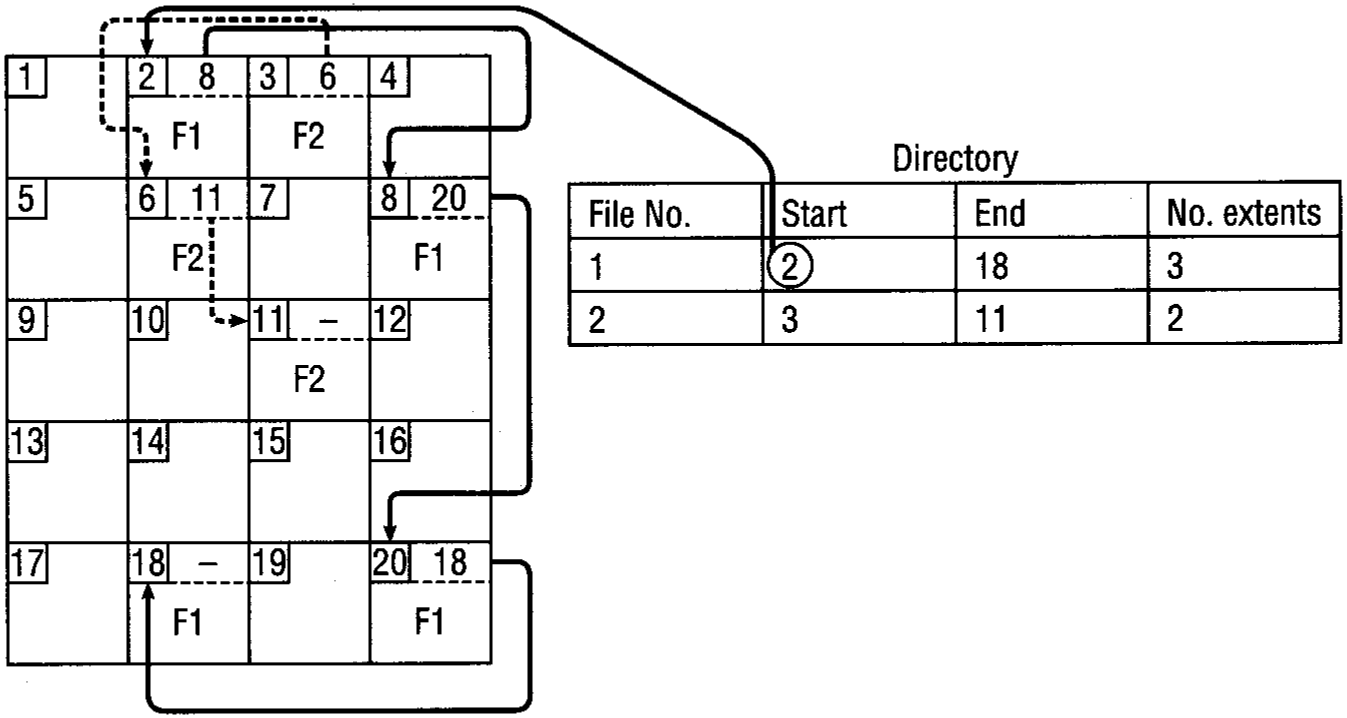
A file may not fill a contiguous run of blocks. This is known as 'fragmentation'.

Noncontiguous Storage (continued)

- **Advantage of noncontiguous storage:**
  - Eliminates external storage fragmentation and need for compaction

  However:
  - Does not support direct access because no easy way to determine exact location of specific record

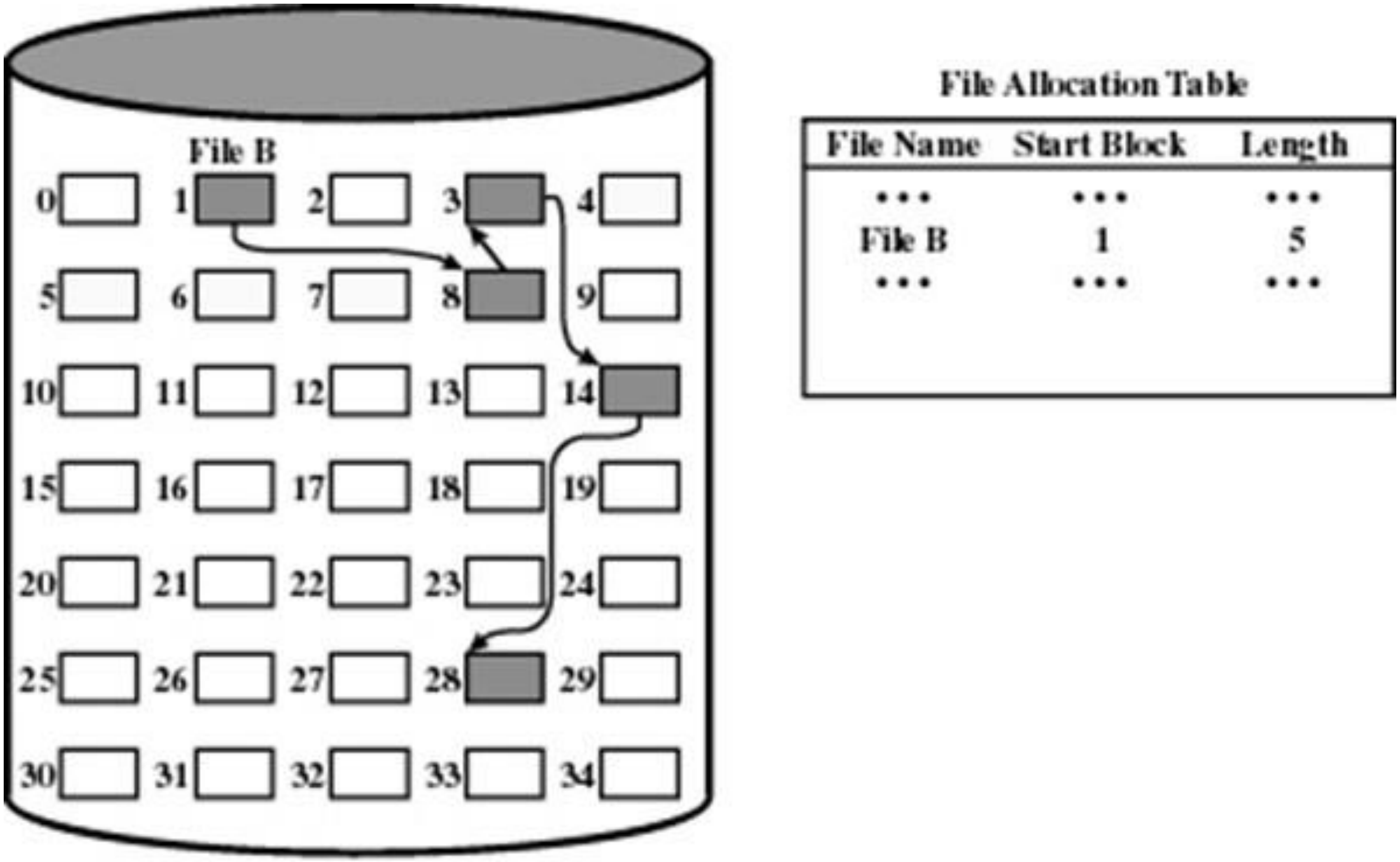
Noncontiguous Storage (continued)



Directory

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

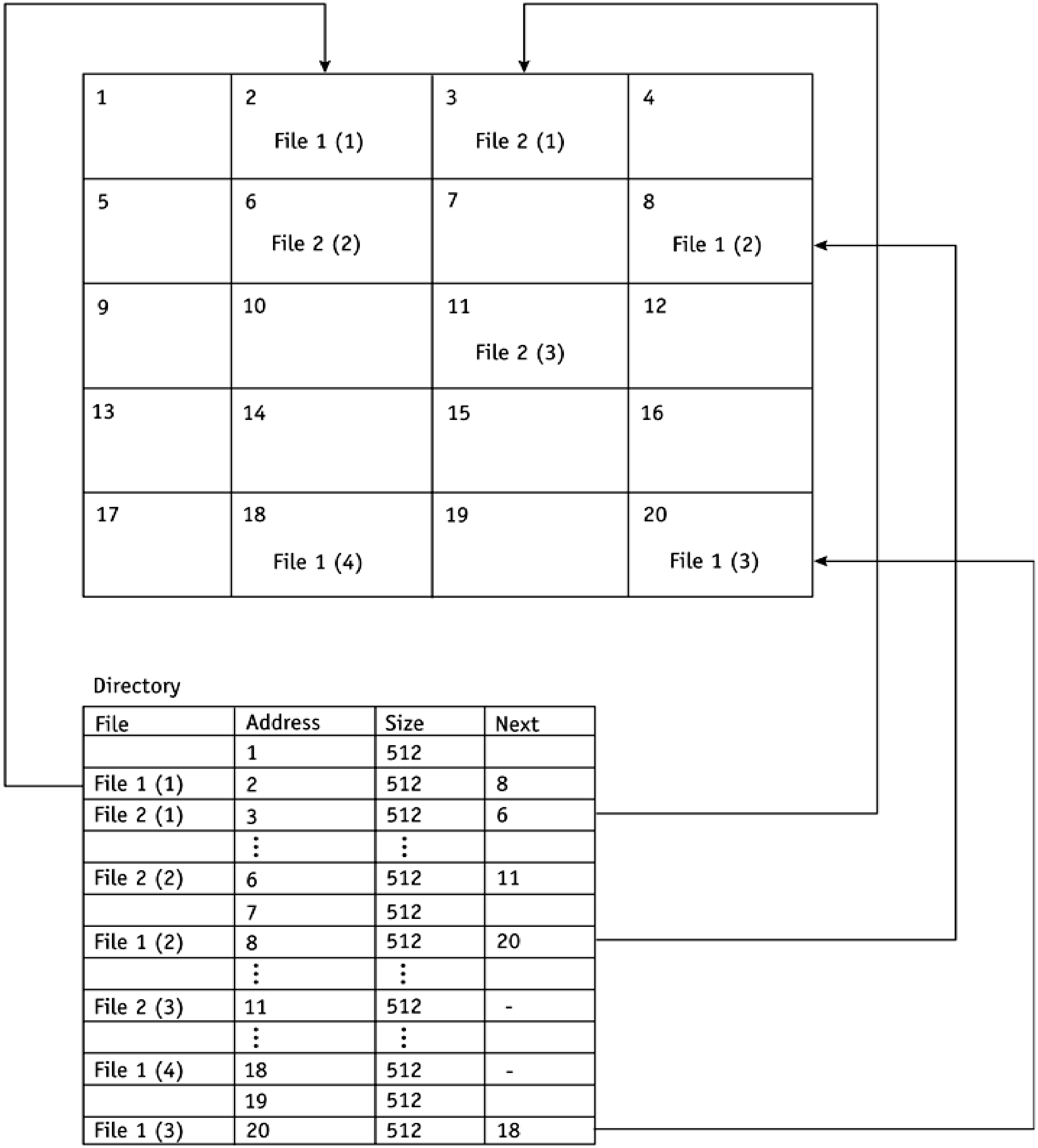
Figure 8.8: Noncontiguous file storage with linking taking place at the storage level



File Allocation Table

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

Figure 12.9 Chained Allocation



1	2 File 1 (1)	3 File 2 (1)	4
5	6 File 2 (2)	7	8 File 1 (2)
9	10	11 File 2 (3)	12
13	14	15	16
17	18 File 1 (4)	19	20 File 1 (3)

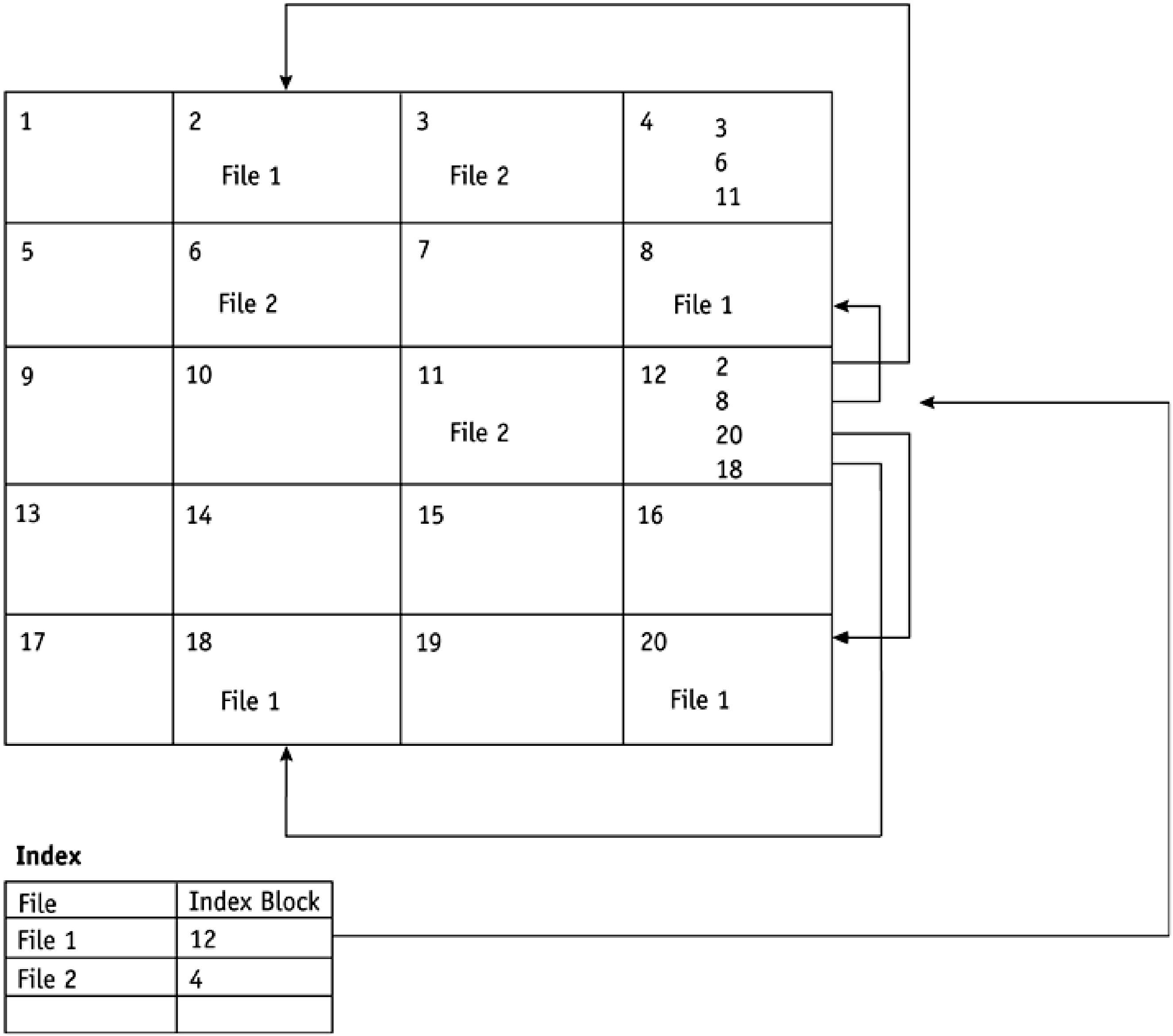
Directory

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 8.9: Noncontiguous file storage with linking taking place at the directory level

Indexed Storage

- Allows direct record access by bringing pointers linking every extent of that file into index block
- Every file has its own index block
  - Consists of addresses of each disk sector that make up the file
  - Lists each entry in the same order in which sectors are linked
- Supports both sequential and direct access
- Doesn’t necessarily improve use of storage space
- Larger files may have several levels of indexes

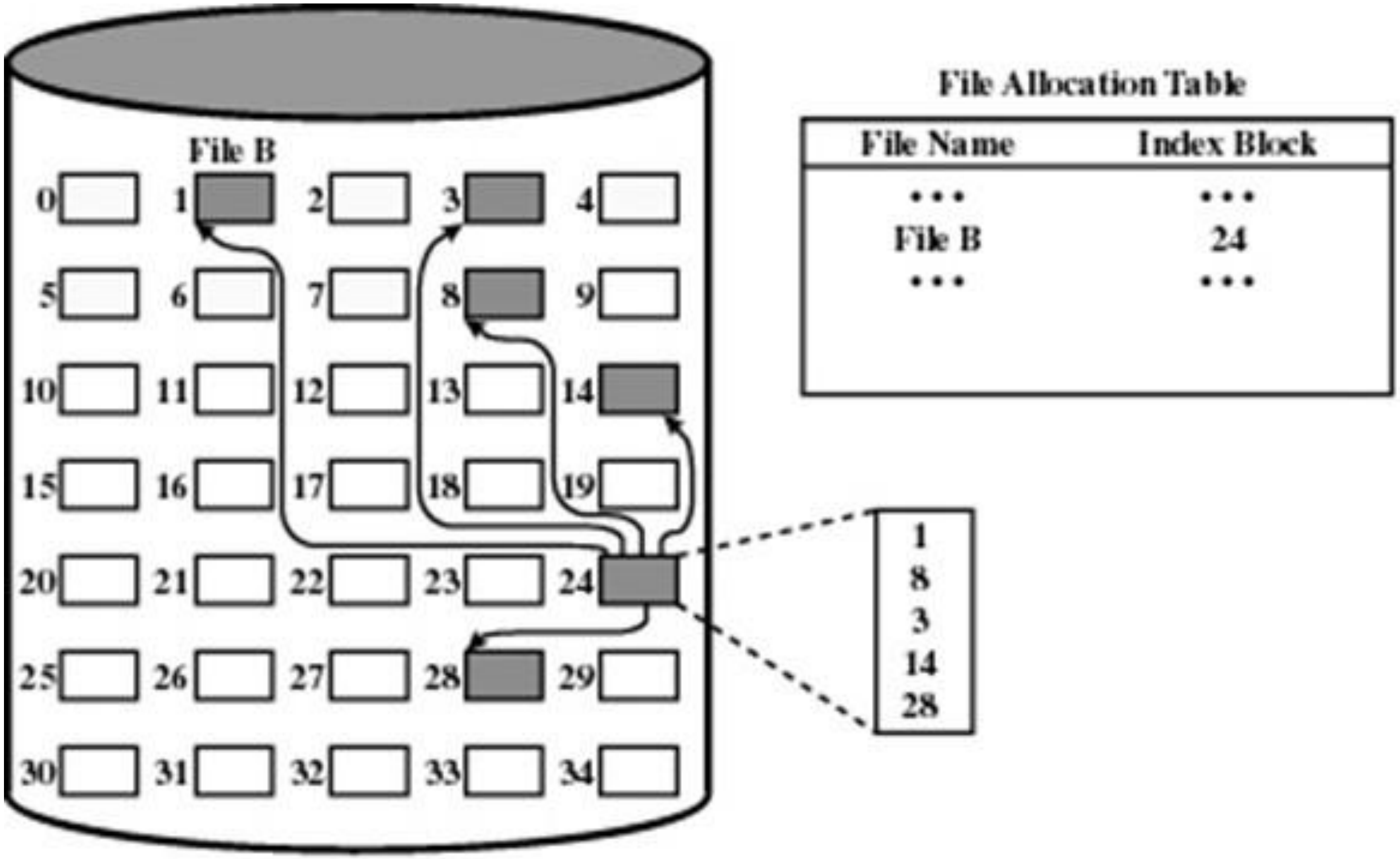


1	2 File 1	3 File 2	4 3 6 11
5	6 File 2	7	8 File 1
9	10	11 File 2	12 2 8 20 18
13	14	15	16
17	18 File 1	19	20 File 1

Index

File	Index Block
File 1	12
File 2	4

Figure 8.10: Indexed storage



File Allocation Table

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

Figure 12.11 Indexed Allocation with Block Portions

Access Methods

- Dictated by a file’s organization
- Most flexibility is allowed with indexed sequential files and least with sequential
- File organized in sequential fashion can support only sequential access to its records
  - Records can be of fixed or variable length
- File Manager uses the address of last byte read to access the next sequential record
- **Current byte address (CBA)** must be updated every time a record is accessed

Access Methods (continued)

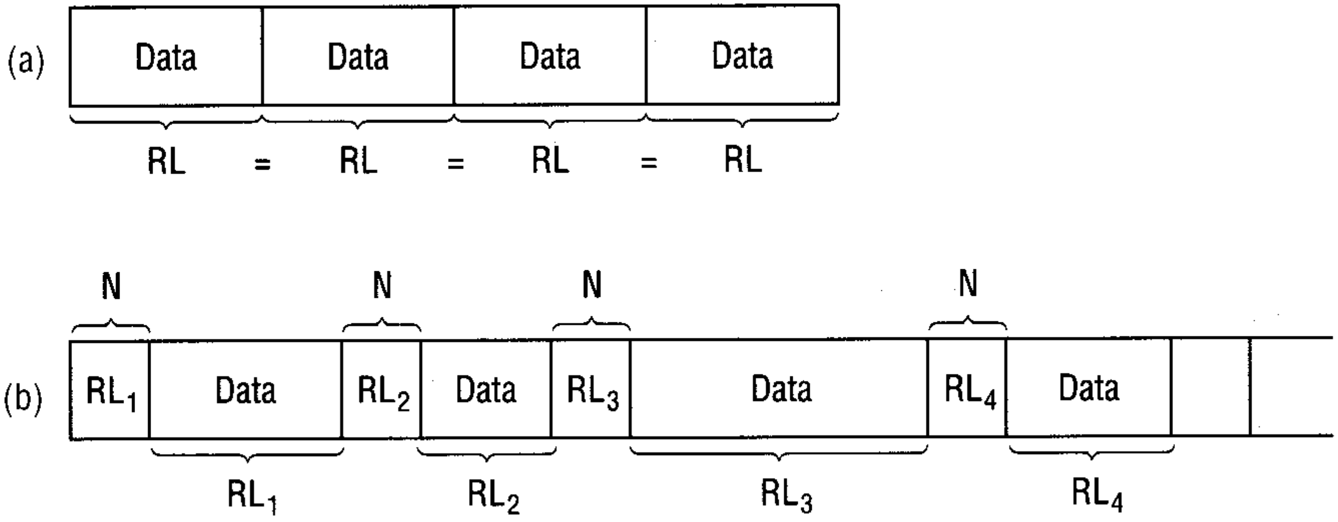


Figure 8.11: (a) Fixed-length records
(b) Variable-length records

Access Methods (continued)

- **Sequential access:**
  - Fixed-length records: (uses Current Byte Address & record length)
    - CBA = CBA + RL
  - Variable-length records:
    - CBA = CBA + N + $RL_{\mathbf{k}}$
- **Direct access:**
  - Fixed-length records:
    - CBA = (RN – 1) * RL; (RN is desired record number)
  - Variable-length records:
    - Virtually impossible because address of desired record can’t be easily computed

Access Methods (continued)

- **Direct access:**
  - Variable-length records: (continued)
    - File Manager must do sequential search through records
    - File Manager can keep table of record numbers and their CBAs (Current Byte Address)
- **Indexed Sequential File:**
  - Can be accessed either sequentially or directly
  - Index file must be searched for the pointer to the block where the data is stored

Levels in a File Management System

- Each level of file management system is implemented by using structured and modular programming techniques
- Each of the modules can be further subdivided into more specific tasks
- Using the information of basic file system, logical file system transforms record number to its byte address
- Verification occurs at every level of the file management system

Levels in a File Management System (continued)

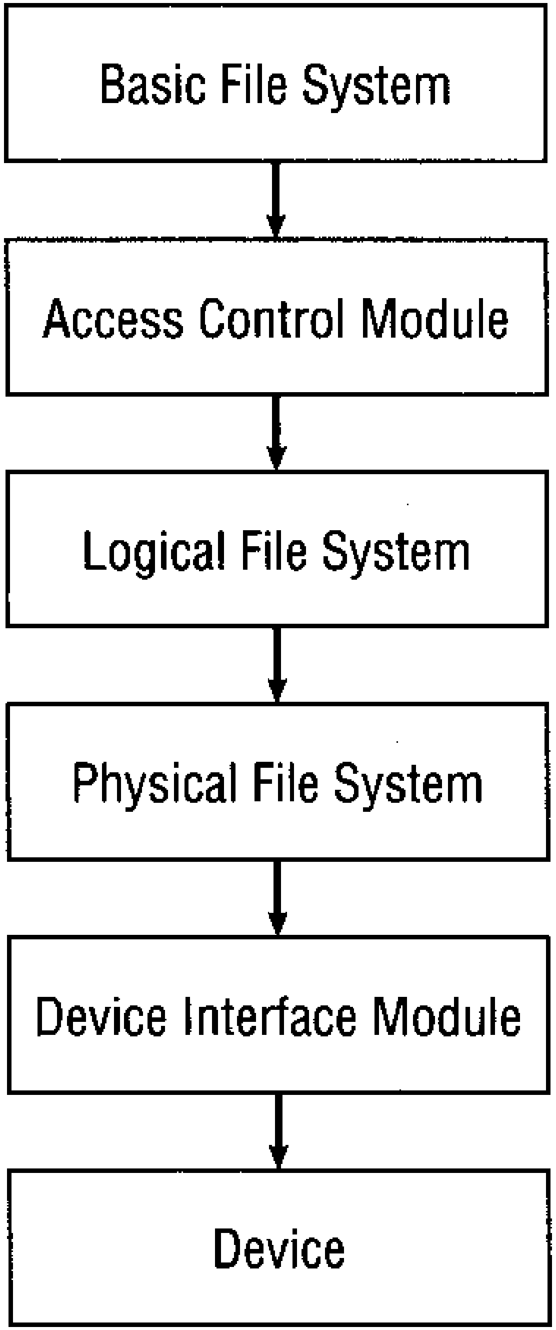


Figure 8.12: File Management System

Levels in a File Management System (continued)

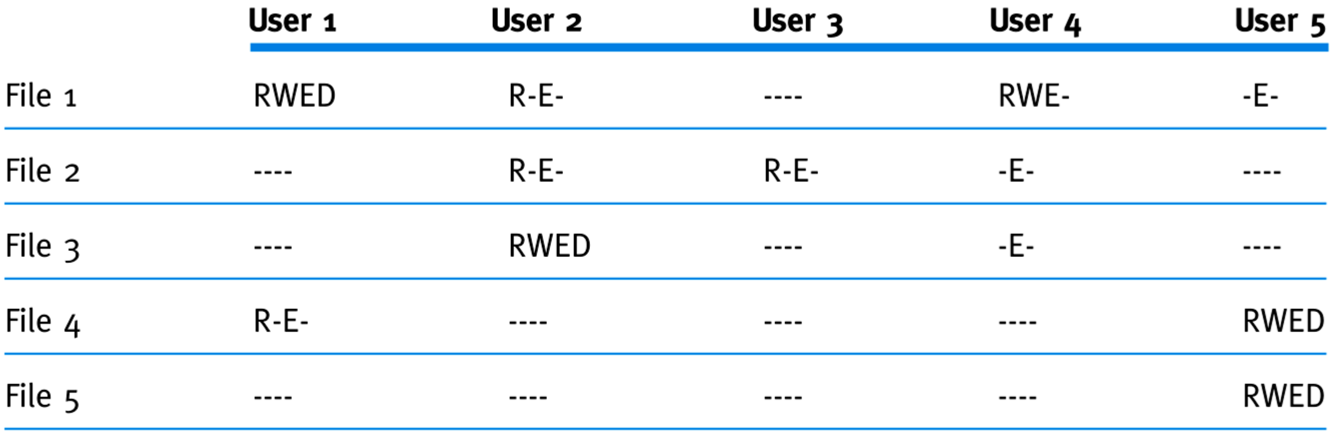
- **Verification** occurs at every level of the file management system:
  - Directory level: file system checks to see if the requested file exists
  - Access control verification module determines whether access is allowed
  - Logical file system checks to see if the requested byte address is within the file's limits
  - Device interface module checks to see whether the storage device exists

Access Control Verification Module

- Each file management system has its own method to control file access
- **Types:**
  - – Access control matrix
  - – Access control lists
  - – Capability lists
  - – Lockword control

Access Control Matrix

- Easy to implement
- Works well for systems with few files & few users
- Results in space wastage because of null entries



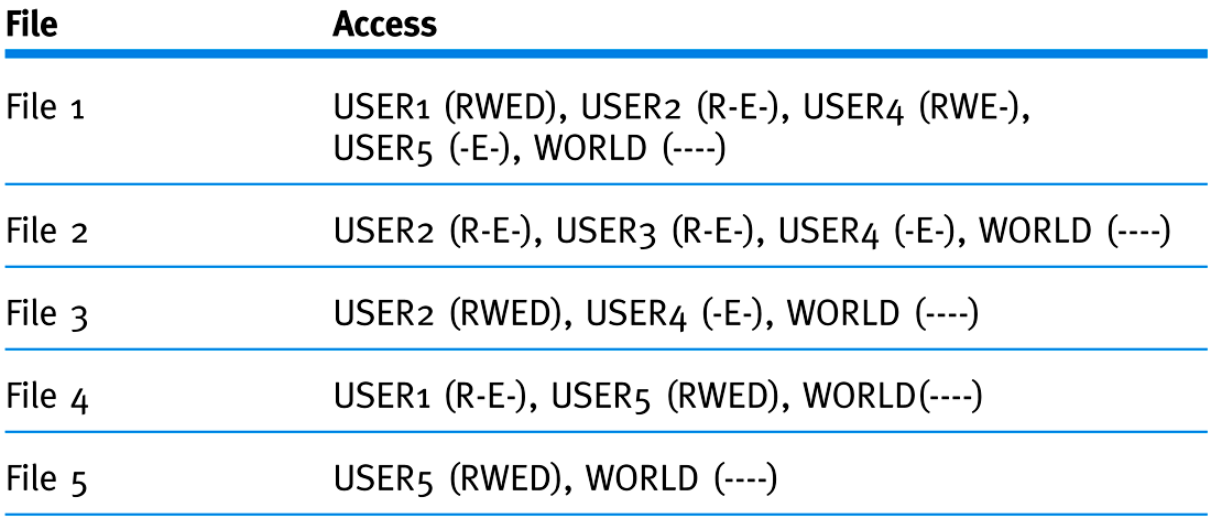
	User 1	User 2	User 3	User 4	User 5
File 1	RWED	R-E-	----	RWE-	-E-
File 2	----	R-E-	R-E-	-E-	----
File 3	----	RWED	----	-E-	----
File 4	R-E-	----	----	----	RWED
File 5	----	----	----	----	RWED

R = Read Access
W = Write Access
E = Execute Access
D = Delete Access
- = Access Not Allowed

Table 8.1: Access Control Matrix

Access Control Lists

- Modification of access control matrix technique
- Each file is entered in list & contains names of users who are allowed access to it and type of access permitted



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 (----)

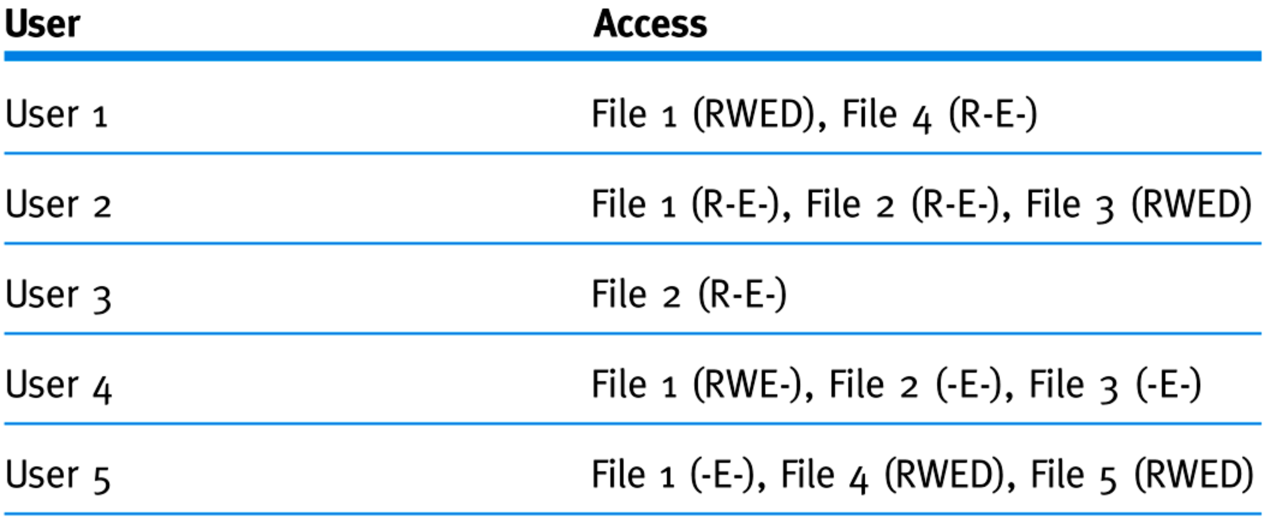
Table 8.2: Access Control List

Access Control Lists (continued)

- Contains the name of only those users who may use file; those denied any access are grouped under “WORLD”
- List is shortened by putting users into categories:
  - **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 the files to which each has access
- Can control access to devices as well as to files



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)

Table 8.3: Capability Lists

Lockwords

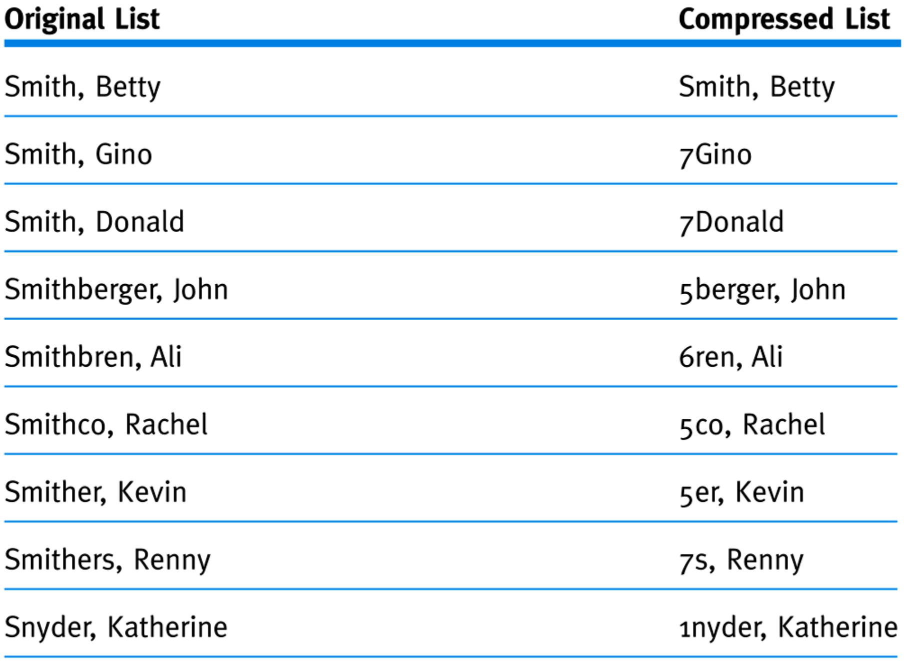
- **Lockword:** similar to a password but protects a single file
- **Advantages:**
  - Requires smallest amount of storage for file protection
- **Disadvantages:**
  - Can be guessed by hackers or passed on to unauthorized users
  - Generally doesn’t control type of access to file
    - Anyone who knows lockword can read, write, execute, or delete file

Data Compression

- A technique used to save space in files
- **Methods for data compression:**
  - **Records with repeated characters:** Repeated characters are replaced with a code
    - e.g., ADAMSbbbbbbbbbb => ADAMSb10
      300000000 => 3#8
  - **Repeated terms:** Compressed by using symbols to represent most commonly used words
    - e.g., in a university's student database common words like student, course, grade, & department could each be represented with single character

Data Compression (continued)

Front-end compression: Each entry takes a given number of characters from the previous entry that they have in common



Original List	Compressed List
Smith, Betty	Smith, Betty
Smith, Gino	7Gino
Smith, Donald	7Donald
Smithberger, John	5berger, John
Smithbren, Ali	6ren, Ali
Smithco, Rachel	5co, Rachel
Smither, Kevin	5er, Kevin
Smithers, Renny	7s, Renny
Snyder, Katherine	1nyder, Katherine

Table 8.4: Front-end compression

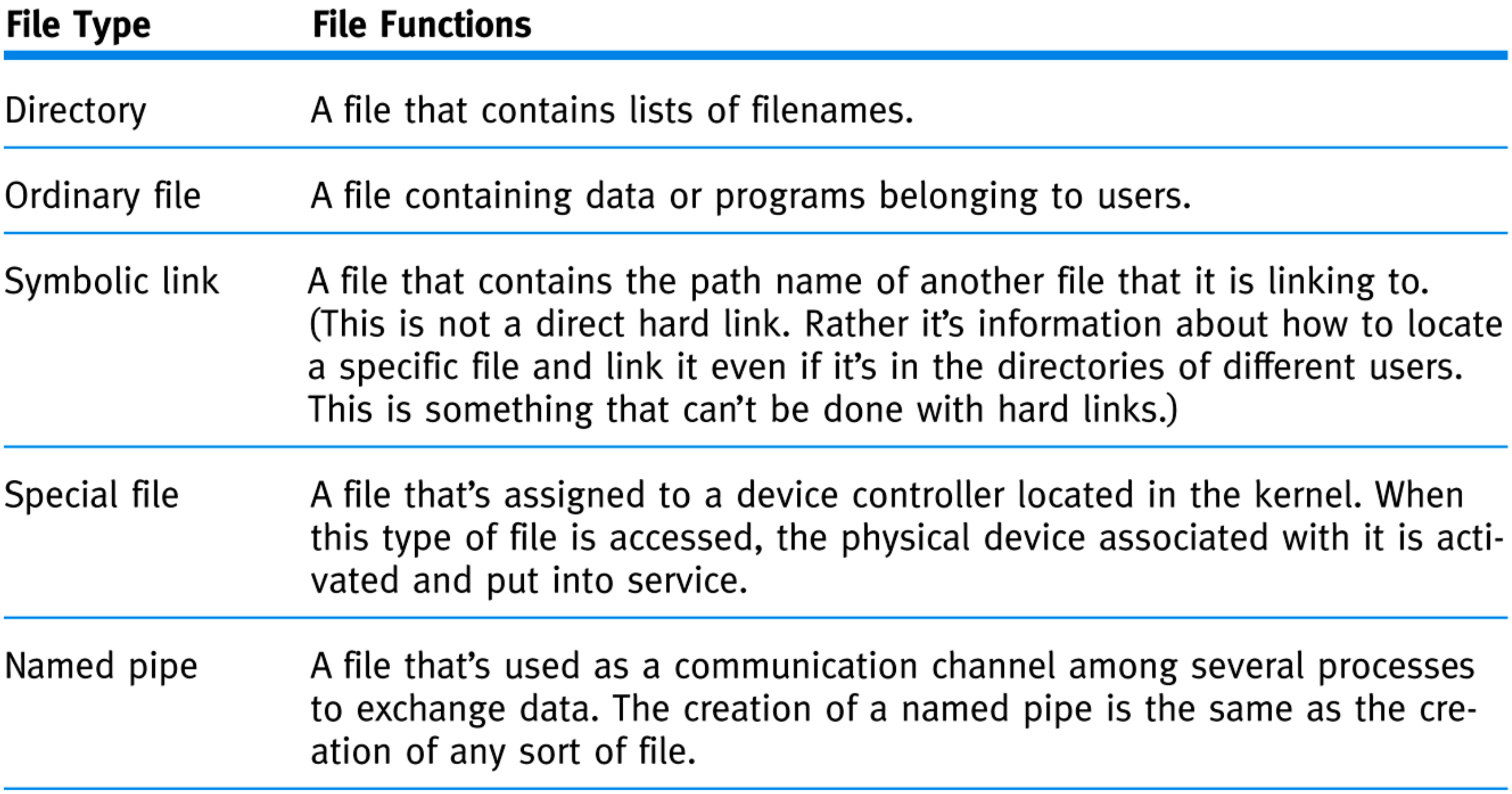
Case Study: File Management in Linux

- All Linux files are organized in directories that are connected to each other in a treelike structure
- Linux specifies five types of files used by the system to determine what the file is to be used for
- **Filenames** can be up to 255 characters long and contain alphabetic characters, underscores, and numbers
- Filename can’t start with a number or a period and can’t contain slashes or quotes

Case Study: File Management in Linux (continued)

- Linux users can obtain **file directories**:
  - By opening the appropriate folder on their desktops
  - Using the command shell interpreter and typing commands after the prompt
- Linux allows three types of file permissions: read (r), write (w), and execute (x)
- **Virtual File System (VFS)** maintains an interface between system calls related to files and the file management code

Case Study: File Management in Linux (continued)



File Type	File Functions
Directory	A file that contains lists of filenames.
Ordinary file	A file containing data or programs belonging to users.
Symbolic link	A file that contains the path name of another file that it is linking to. (This is not a direct hard link. Rather it's information about how to locate a specific file and link it even if it's in the directories of different users. This is something that can't be done with hard links.)
Special file	A file that's assigned to a device controller located in the kernel. When this type of file is accessed, the physical device associated with it is activated and put into service.
Named pipe	A file that's used as a communication channel among several processes to exchange data. The creation of a named pipe is the same as the creation of any sort of file.

Table 8.5: Types of Linux files

Summary

- The File Manager controls every file in the system
- Processes user commands (read, write, modify, create, delete, etc.) to interact with any other file
- Manages access control procedures to maintain the integrity and security of the files under its control
- File Manager must accommodate a variety of file organizations, physical storage allocation schemes, record types, and access methods

Summary (continued)

- Each level of file management system is implemented with structured and modular programming techniques
- Verification occurs at every level of the file management system
- Data compression saves space in files
- Linux specifies five types of files used by the system
- VFS maintains an interface between system calls related to files and the file management code