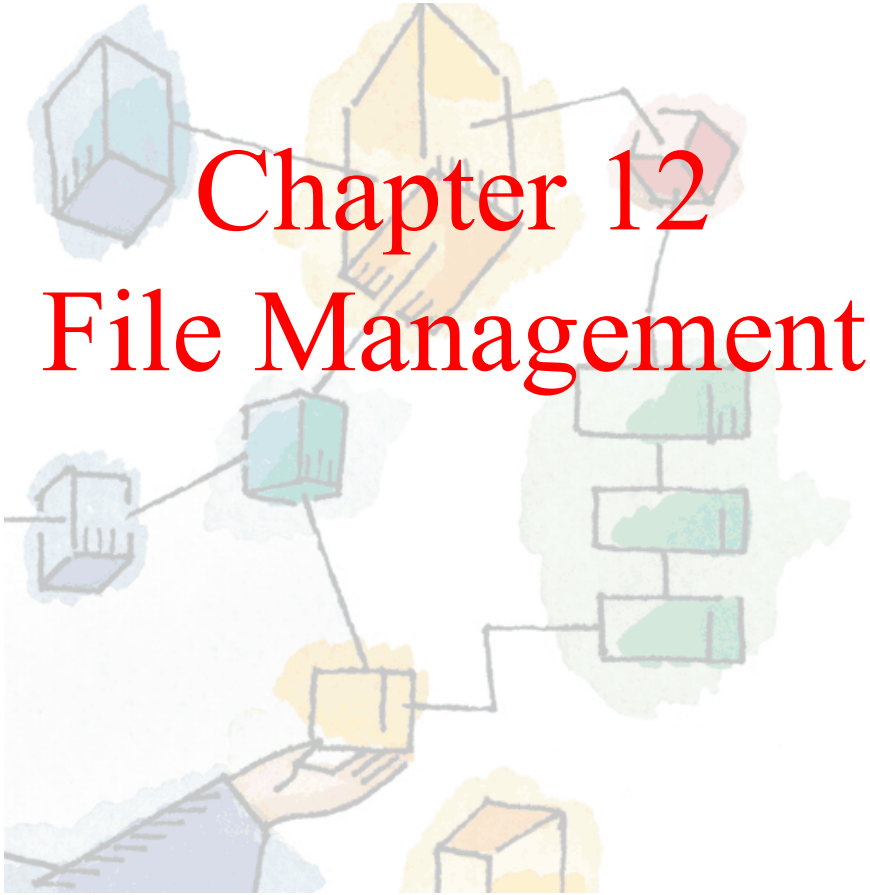
Operating Systems:
Internals and Design Principles, 6/E
William Stallings



Chapter 12
File Management

Patricia Roy
Manatee Community College, Venice, FL

File Management

- File management system consists of system utility programs that run as privileged applications (on behalf of users)
- Concerned with secondary storage
- Long-term existence
- Shared between processes
- Structure (internal or directories)

Win32 API function	UNIX	Description
CreateFile	open	Create a file or open an existing file; return a handle
DeleteFile	unlink	Destroy an existing file
CloseHandle	close	Close a file
ReadFile	read	Read data from a file
WriteFile	write	Write data to a file
SetFilePointer	lseek	Set the file pointer to a specific place in the file
GetFileAttributes	stat	Return the file properties
LockFile	fcntl	Lock a region of the file to provide mutual exclusion
UnlockFile	fcntl	Unlock a previously locked region of the file

- Principle Win32 API functions for file I/O
- Second column gives nearest UNIX equivalent

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

Possible file attributes

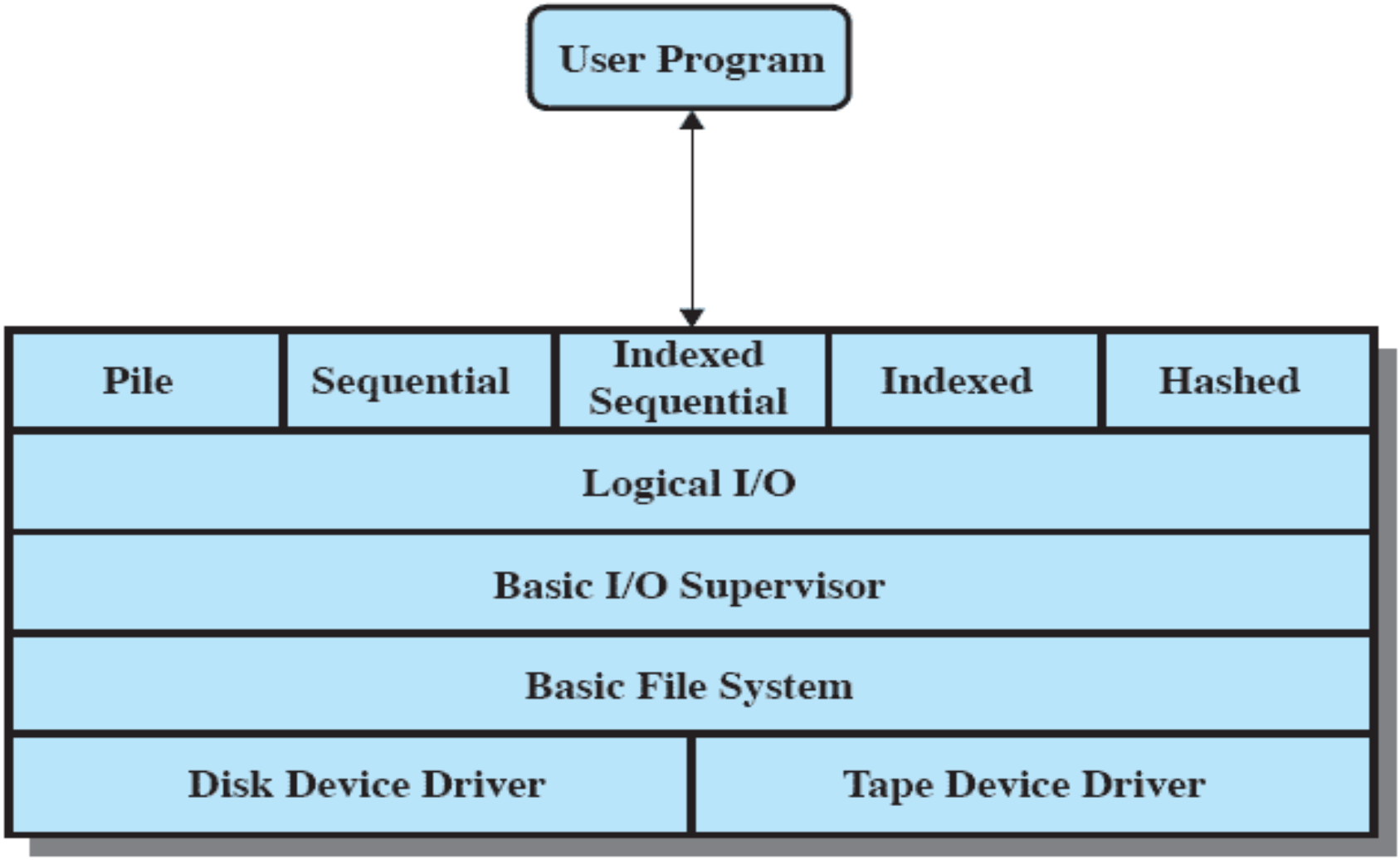


Figure 12.1 File System Software Architecture

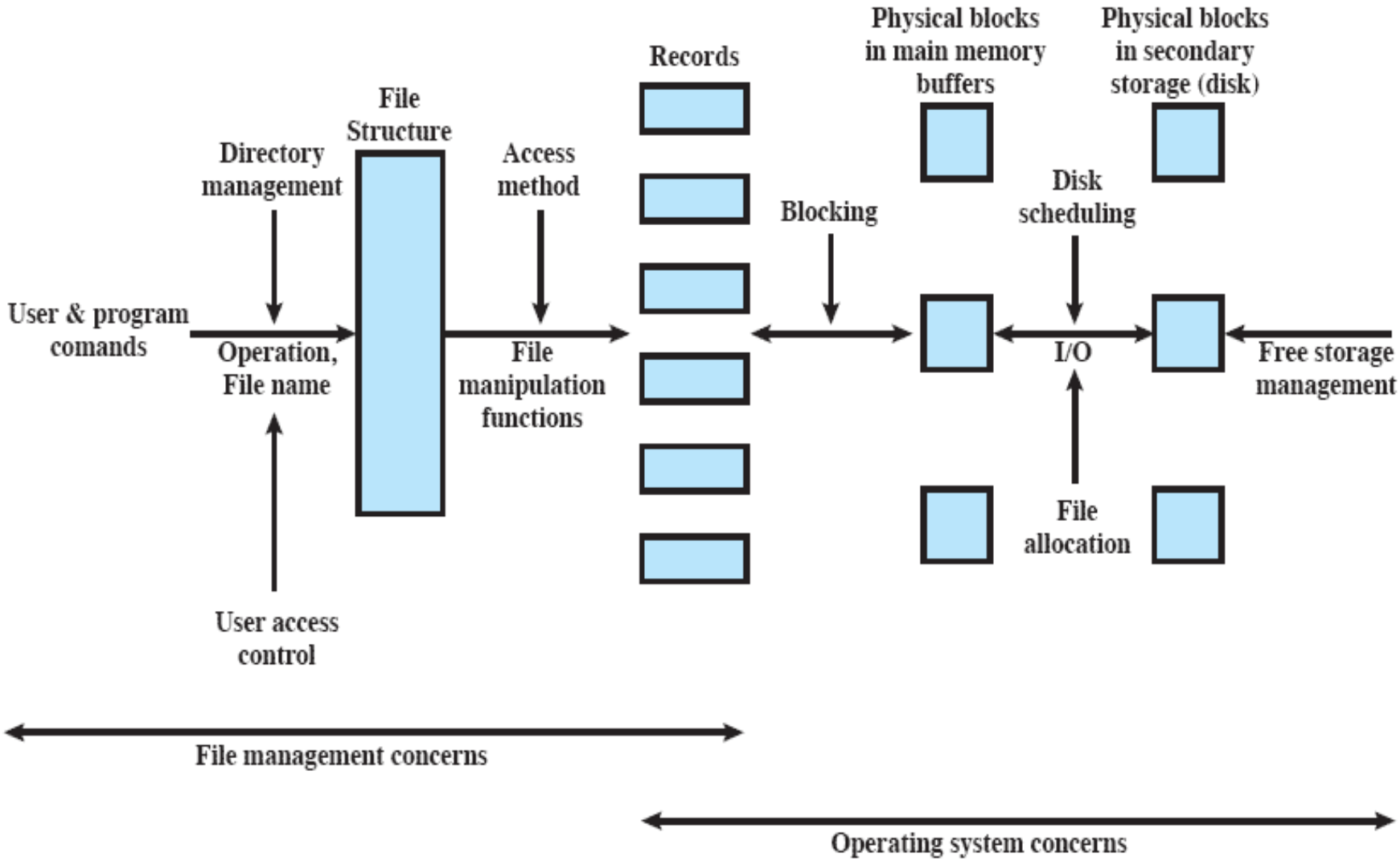


Figure 12.2 Elements of File Management

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

Criteria for File Organization

- Quick random access
  - Needed when accessing a single record
- Ease of update
- 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

Directory System Calls

Win32 API function	UNIX	Description
CreateDirectory	mkdir	Create a new directory
RemoveDirectory	rmdir	Remove an empty directory
FindFirstFile	opendir	Initialize to start reading the entries in a directory
FindNextFile	readdir	Read the next directory entry
MoveFile	rename	Move a file from one directory to another
SetCurrentDirectory	chdir	Change the current working directory

- Principle Win32 API functions for directory management
- Second column gives nearest UNIX equivalent, when one exists

Hierarchical, or Tree-Structured Directory

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

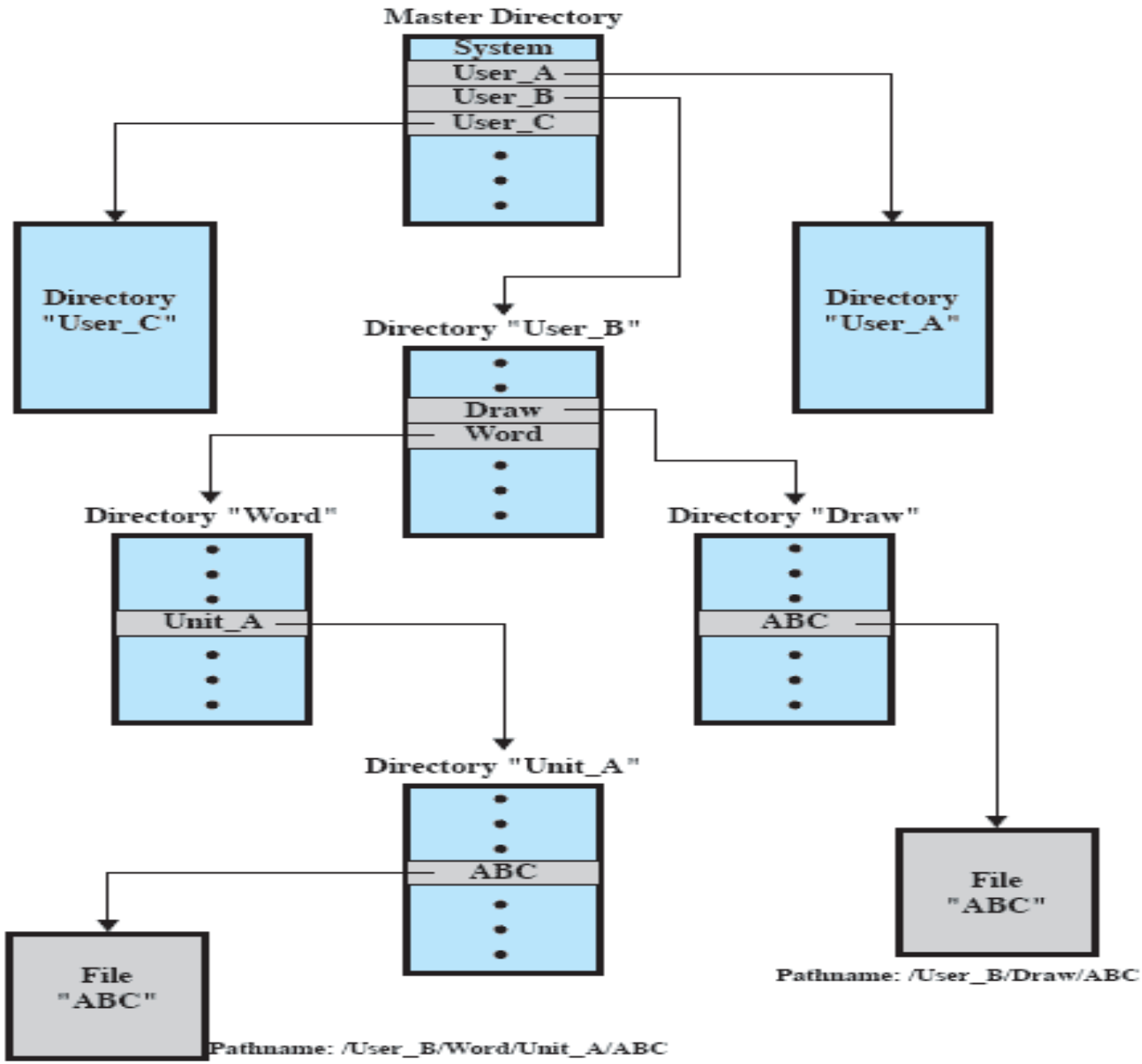


Figure 12.5 Example of Tree-Structured Directory

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 **absolute 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 – **relative pathname**

File Sharing

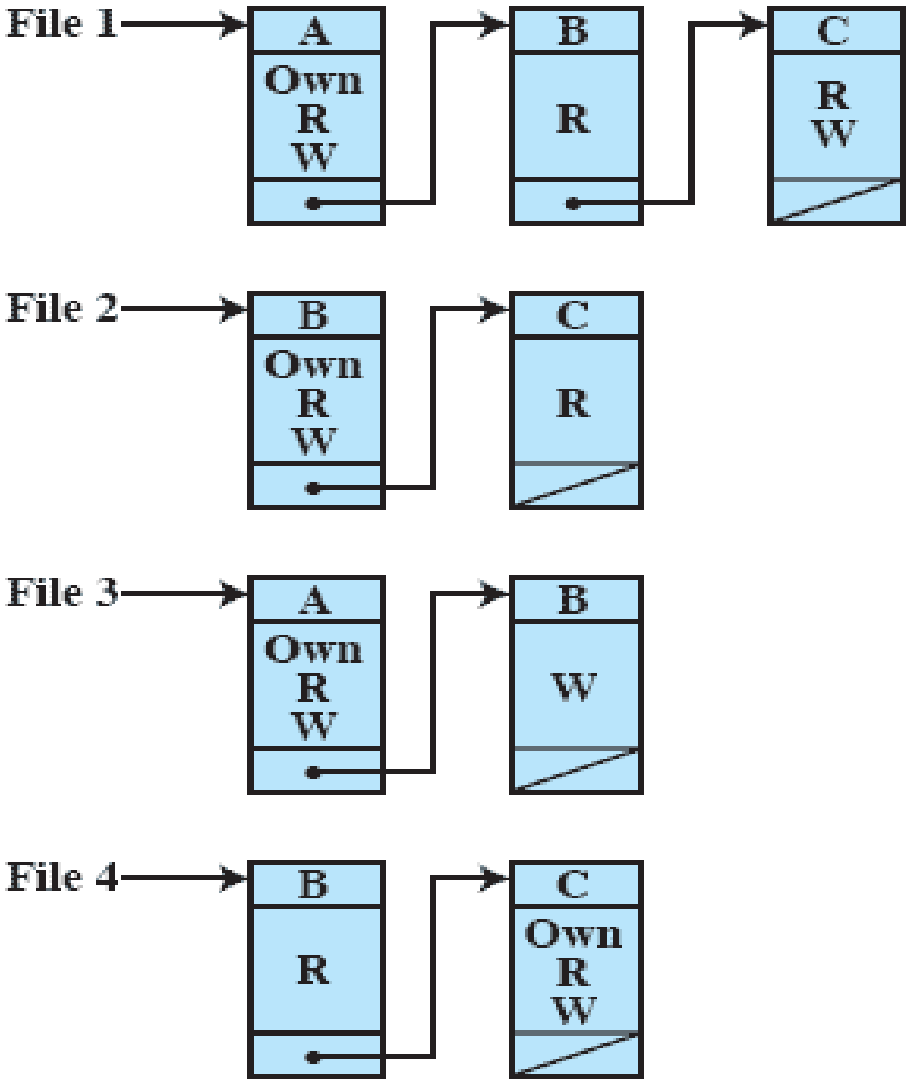
- In multi-user system, allow files to be shared among users
- Two issues
  - Access rights (read, write, execute)
  - Management of simultaneous access (cf. interprocess communication, mutual exclusion)

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

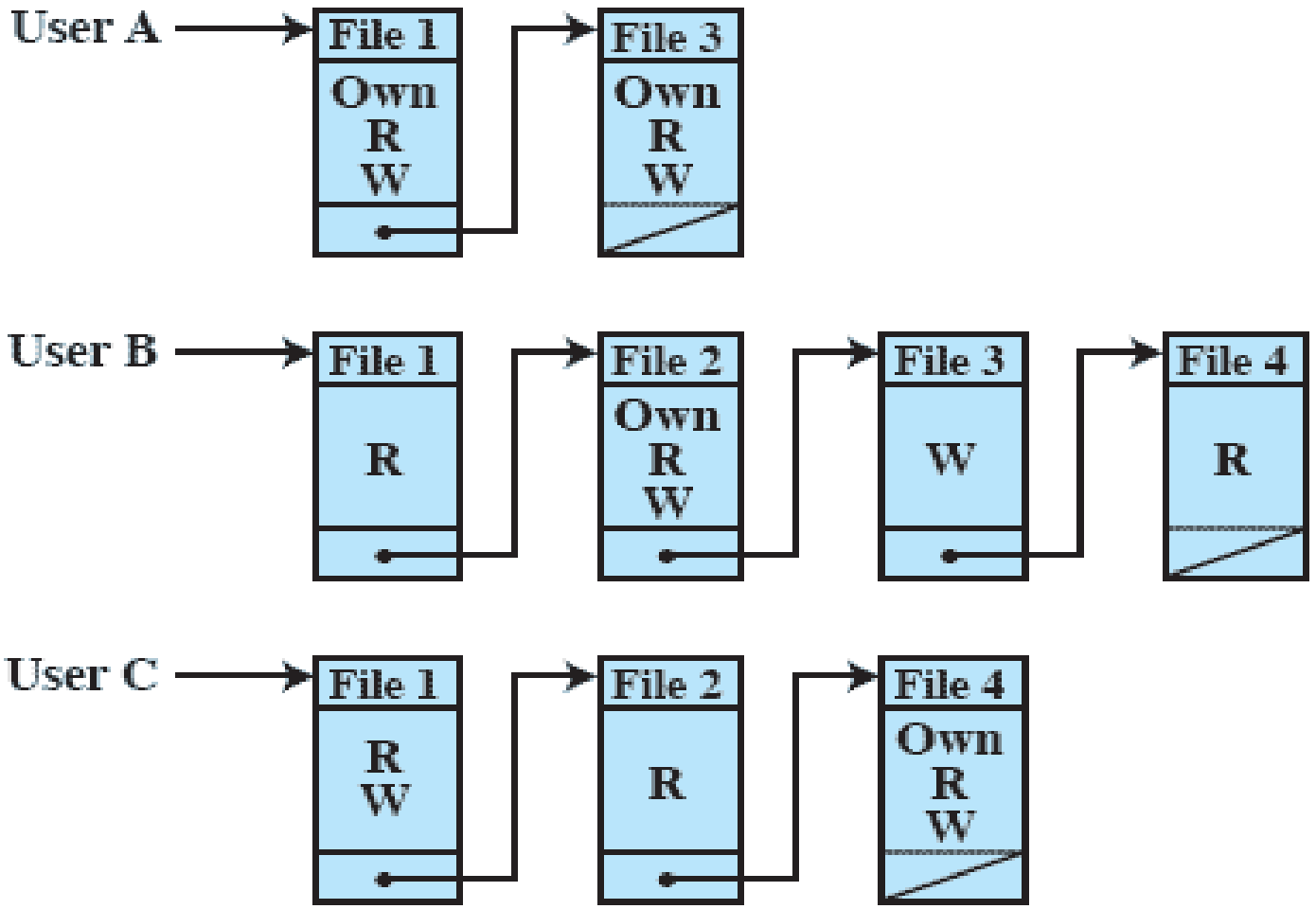
(a) Access matrix

Access Control List



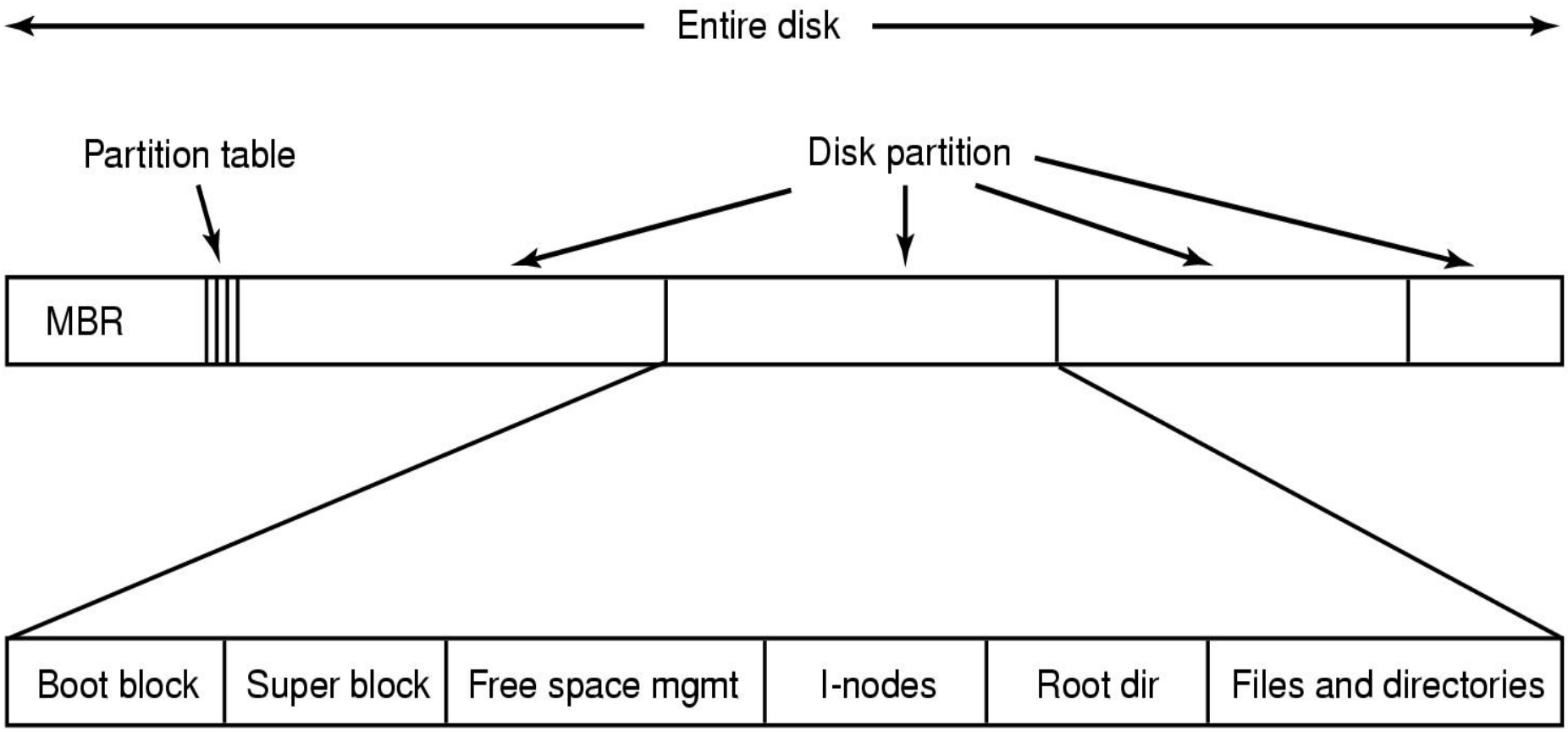
(b) Access control lists for files of part (a)

Capability Lists



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

File System Implementation



A possible file system layout

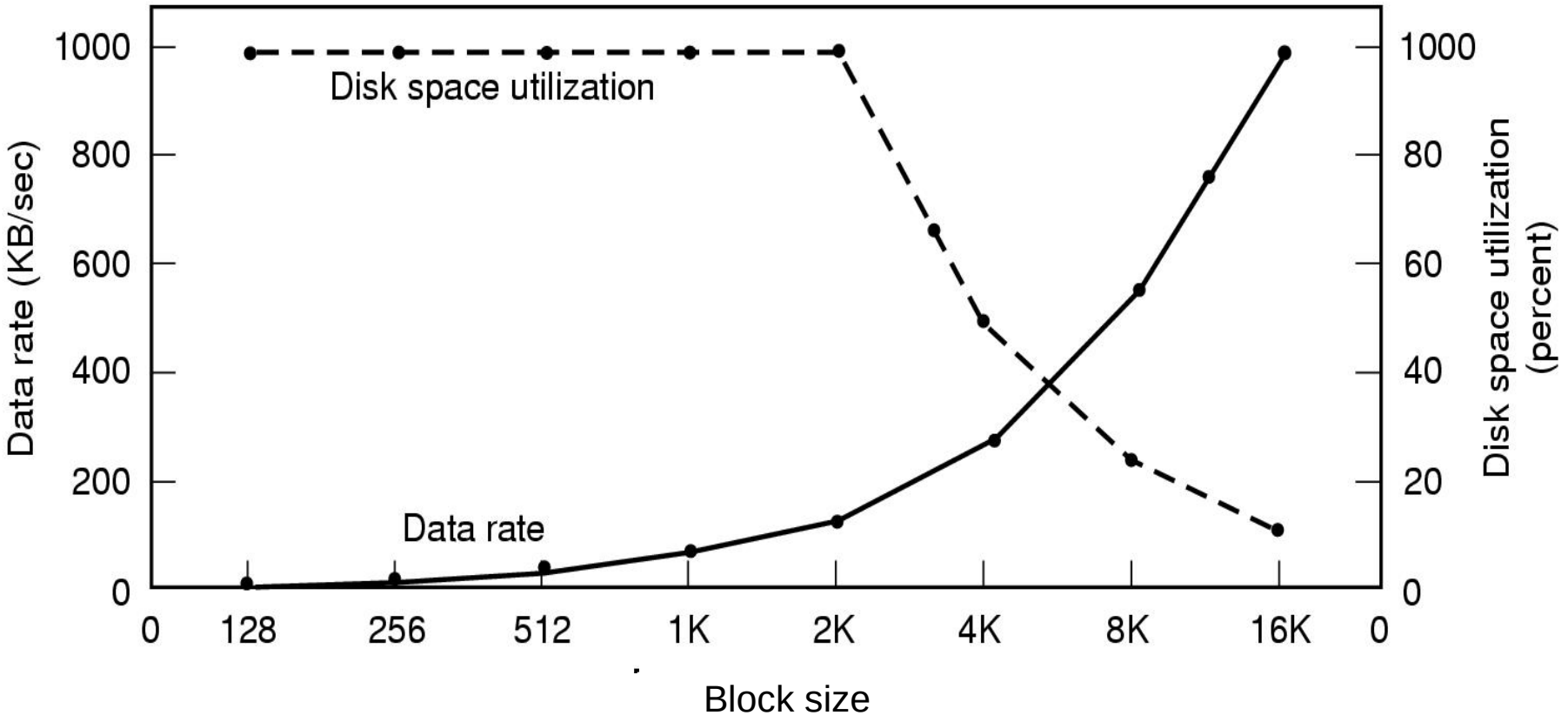
Secondary Storage Management

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

  size of blocks

  allocation methods

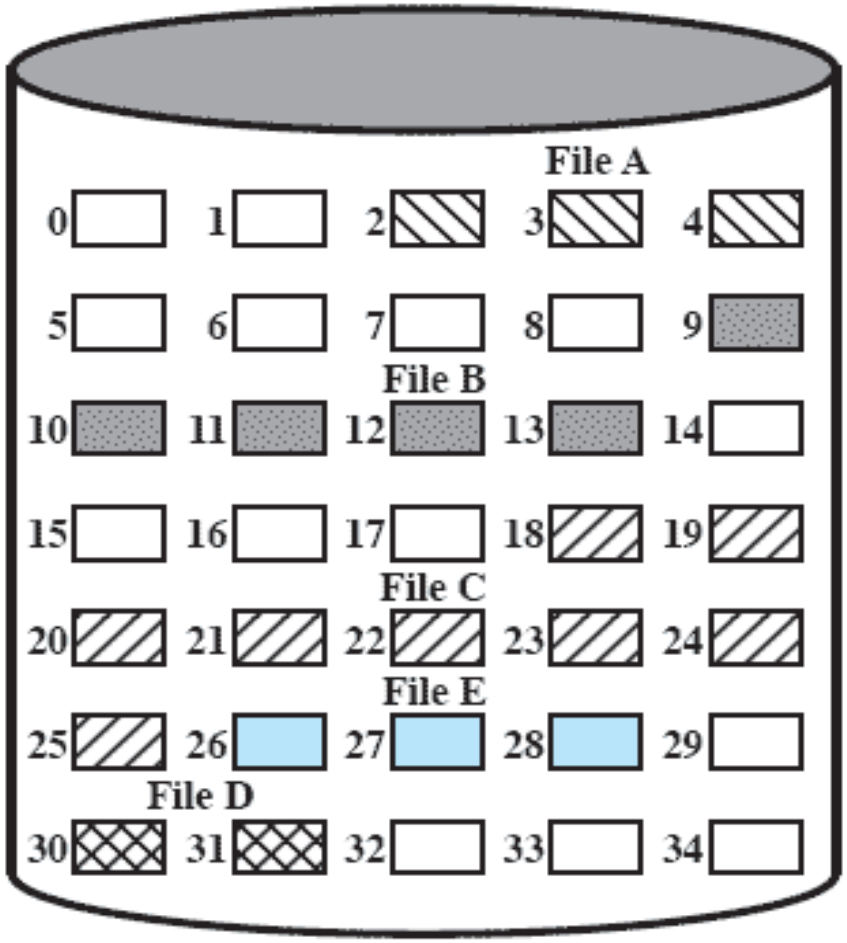
Disk Space Management



- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB

Contiguous Allocation

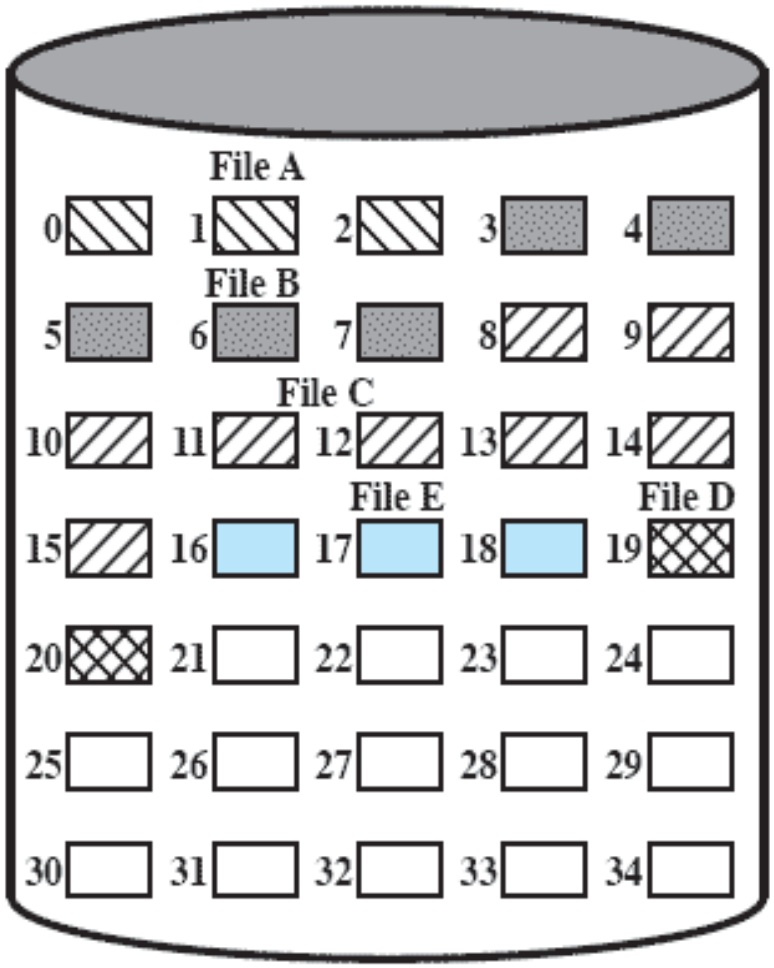
- Single set of blocks is allocated to a file at the time of creation
- Only a single entry in the file allocation table
  - Starting block and length of the file
- External fragmentation will occur
  - Need to perform compaction



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

Figure 12.7 Contiguous File Allocation



File Allocation Table

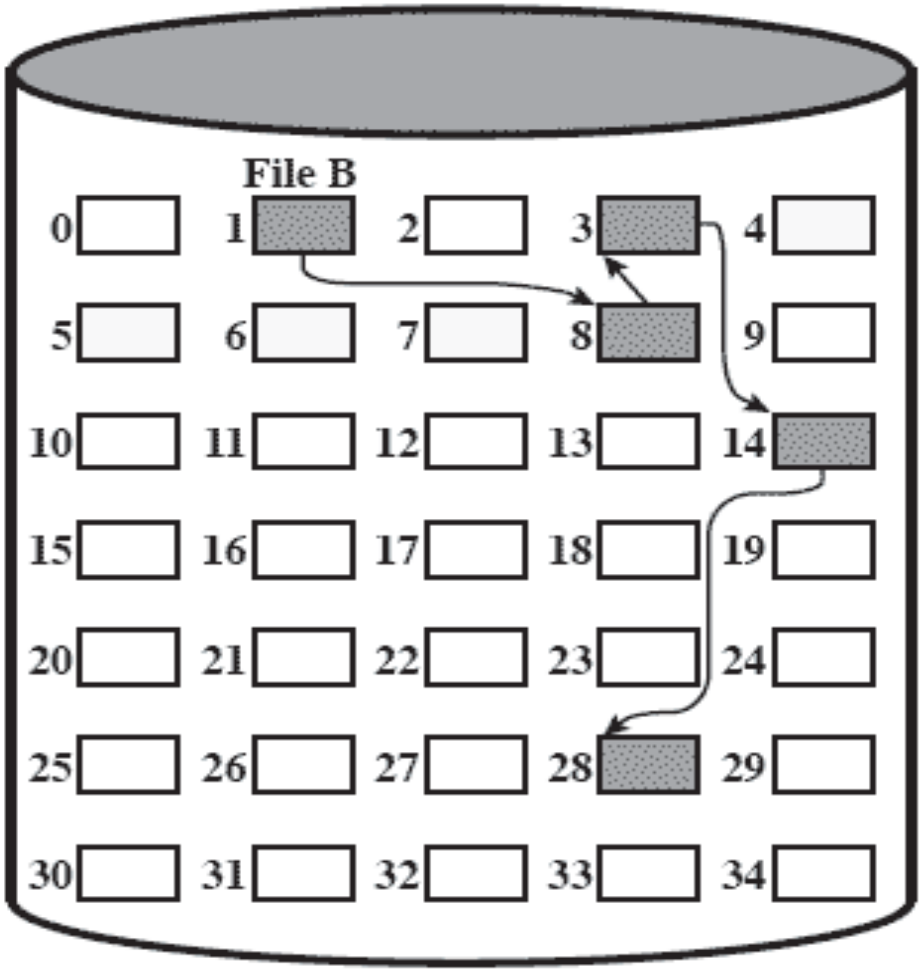
File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

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
- No external fragmentation
- Best for sequential files

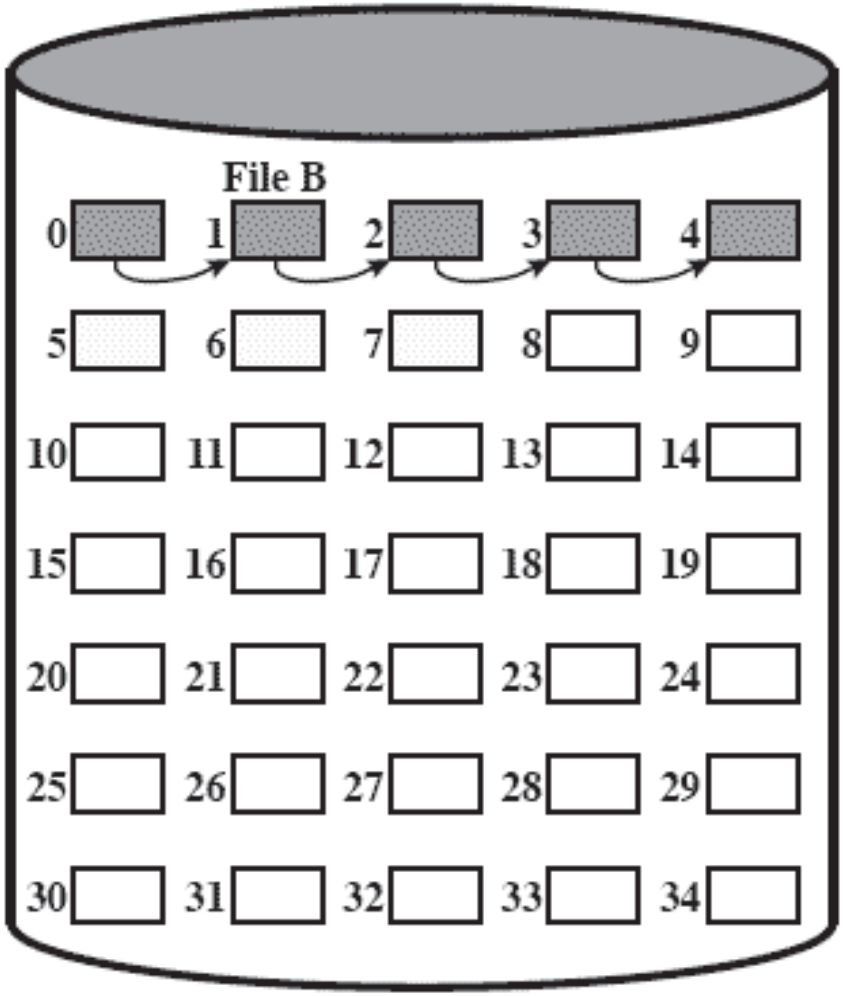
No accommodation of the principle of locality



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)

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

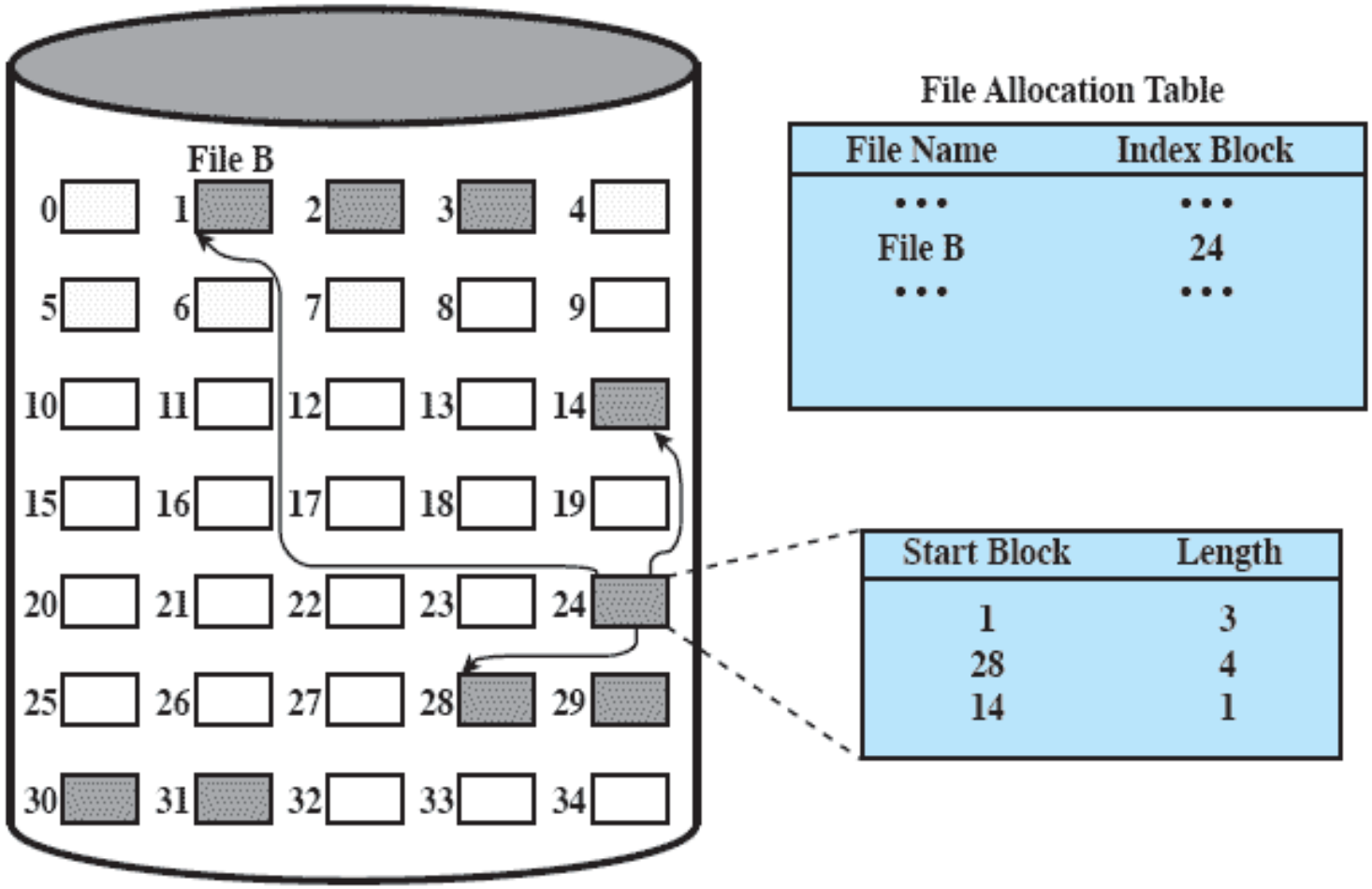
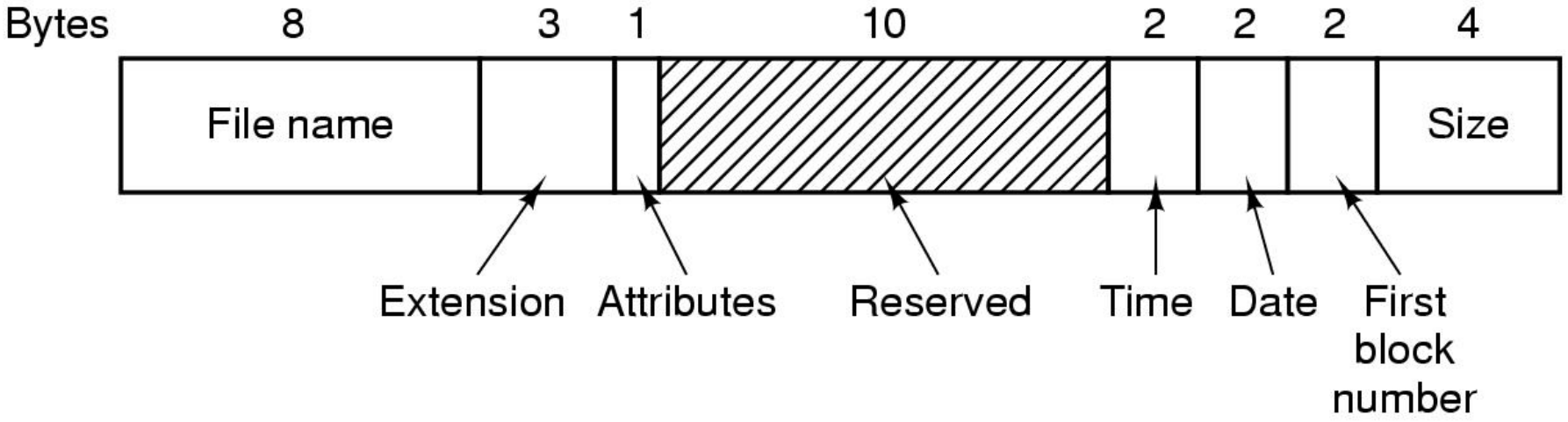


Figure 12.12 Indexed Allocation with Variable-Length Portions

The MS-DOS File System (1)



The MS-DOS directory entry

The MS-DOS File System (2)

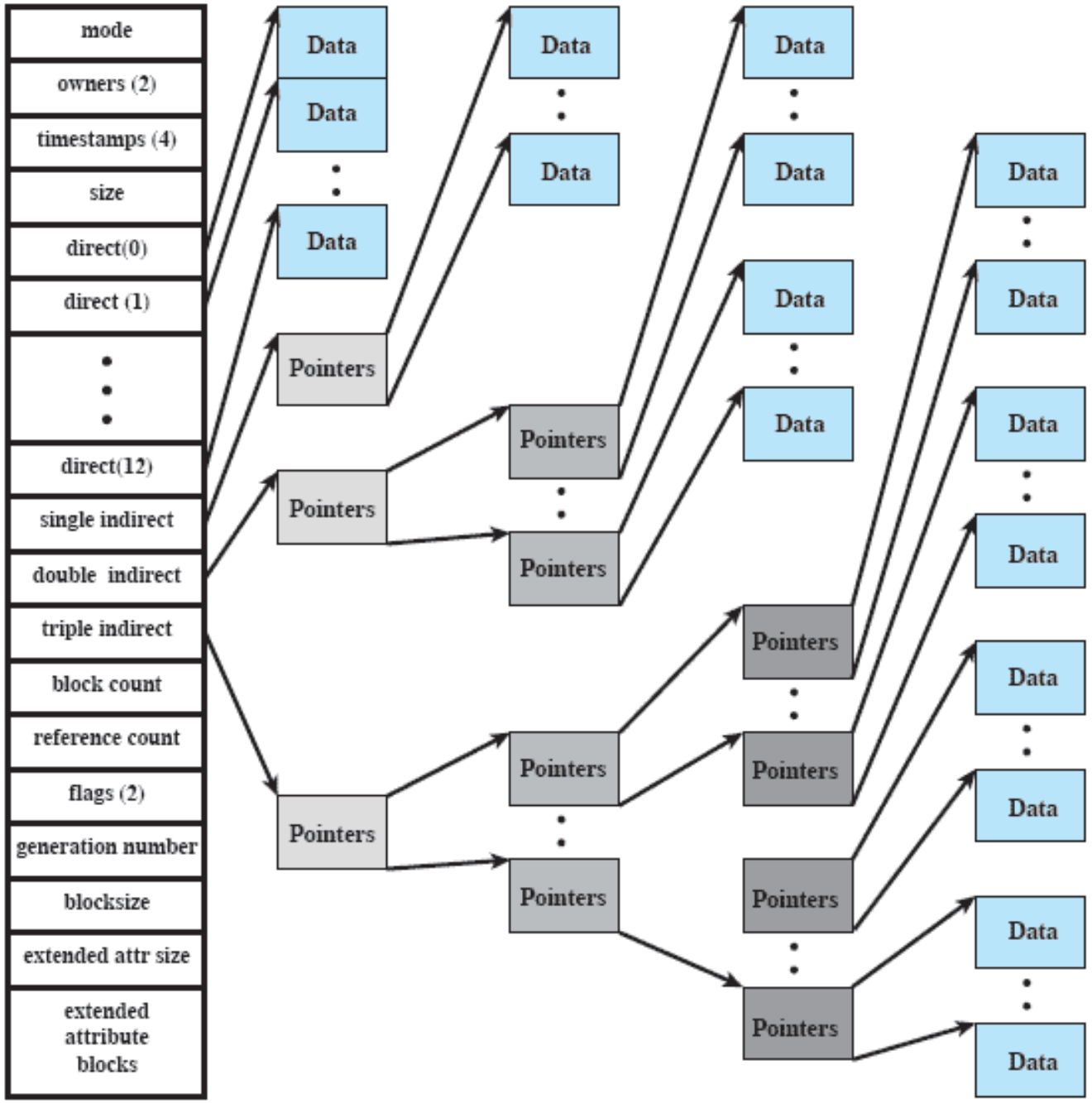
Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

I-nodes

- Index node
- Control structure that contains key information for a particular file: attributes and location of blocks
- Direct and indirect blocks

FreeBSD File Allocation



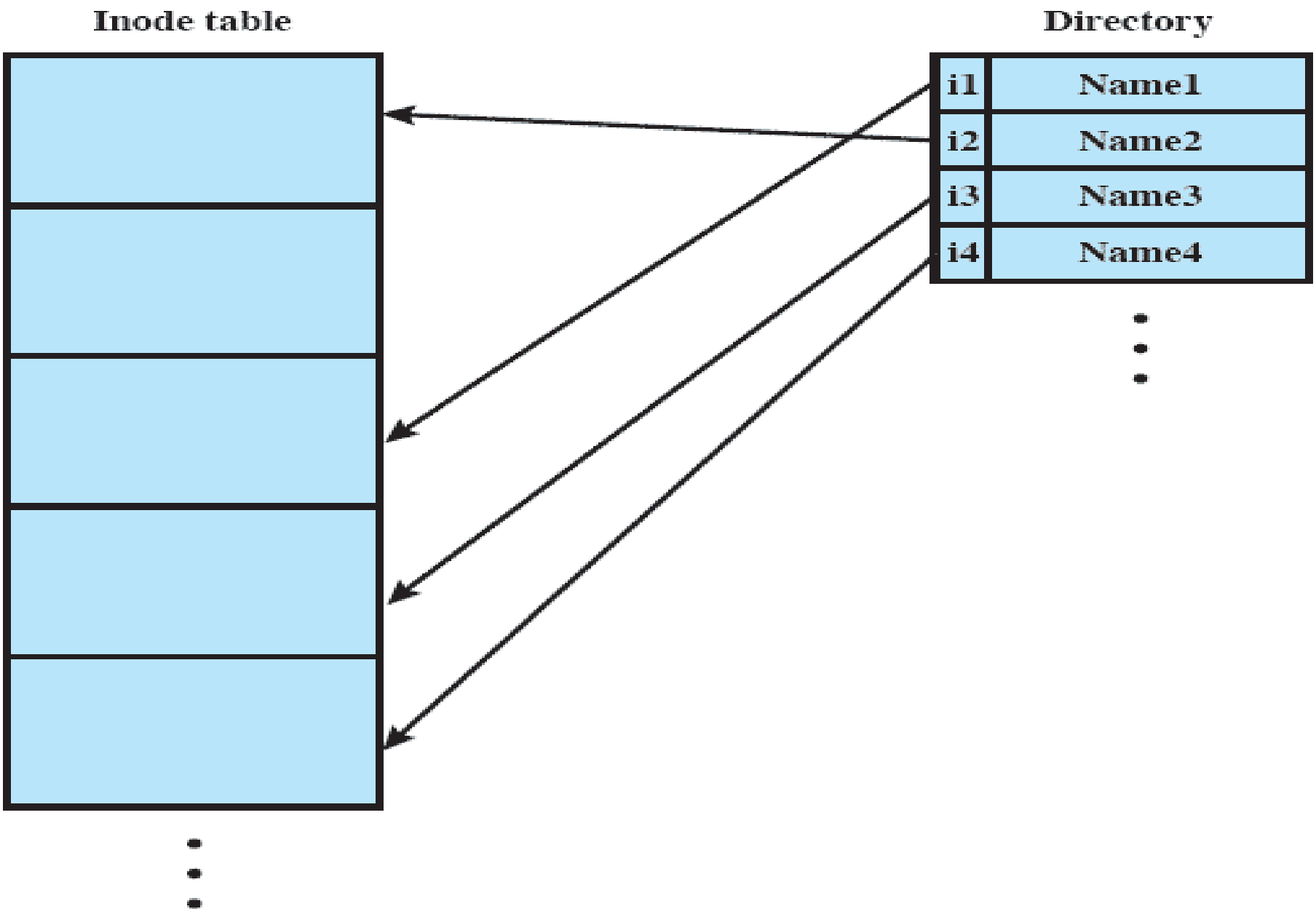
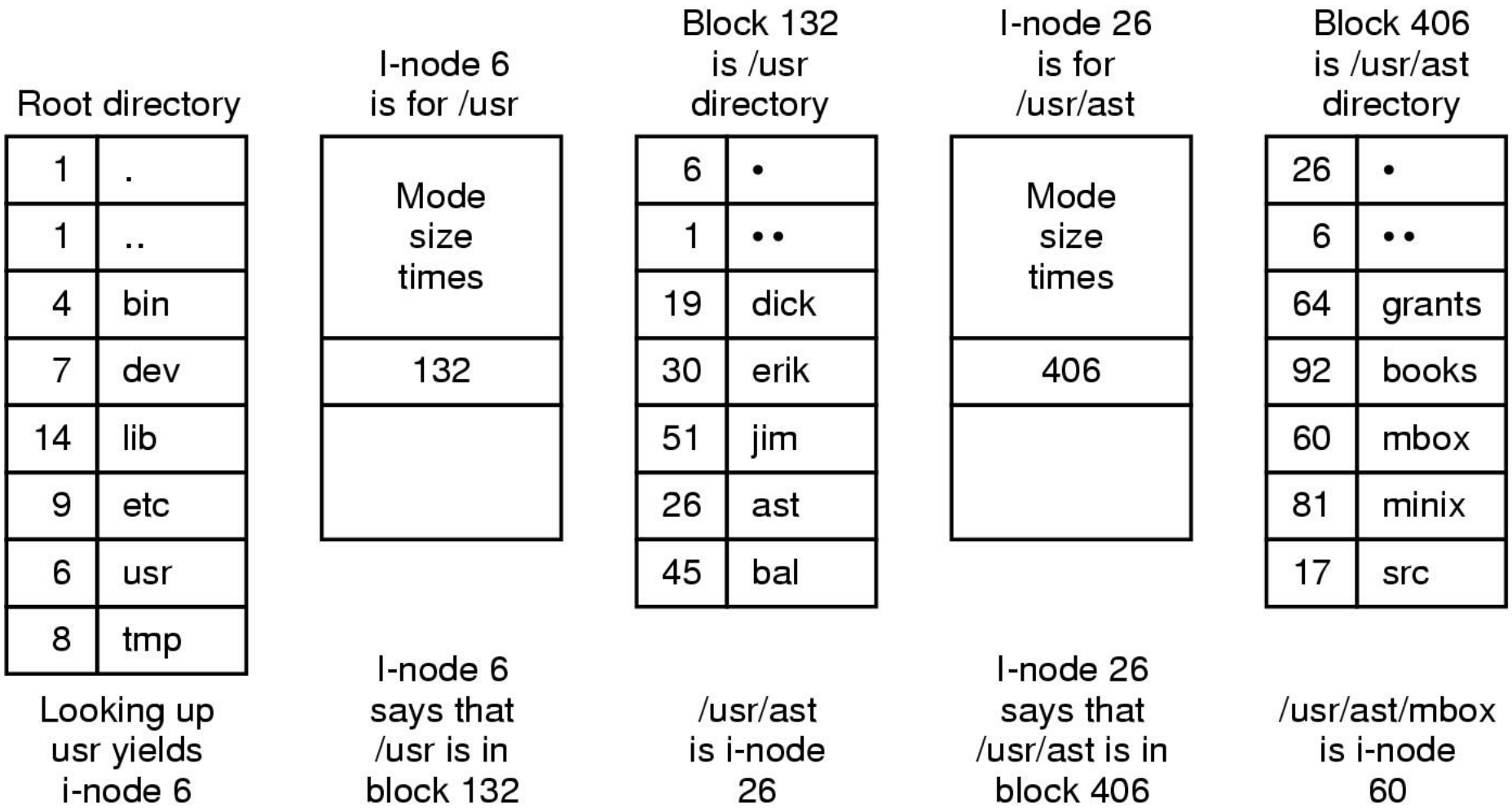


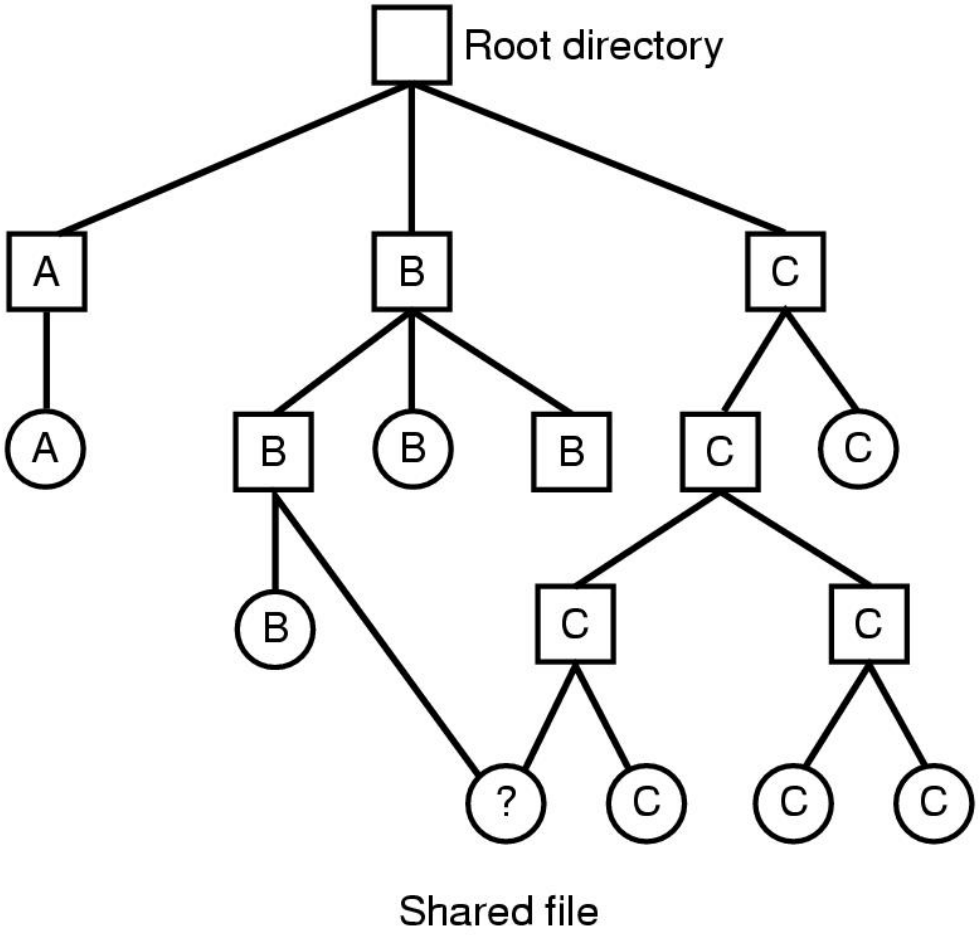
Figure 12.15 UNIX Directories and Inodes

The UNIX File System



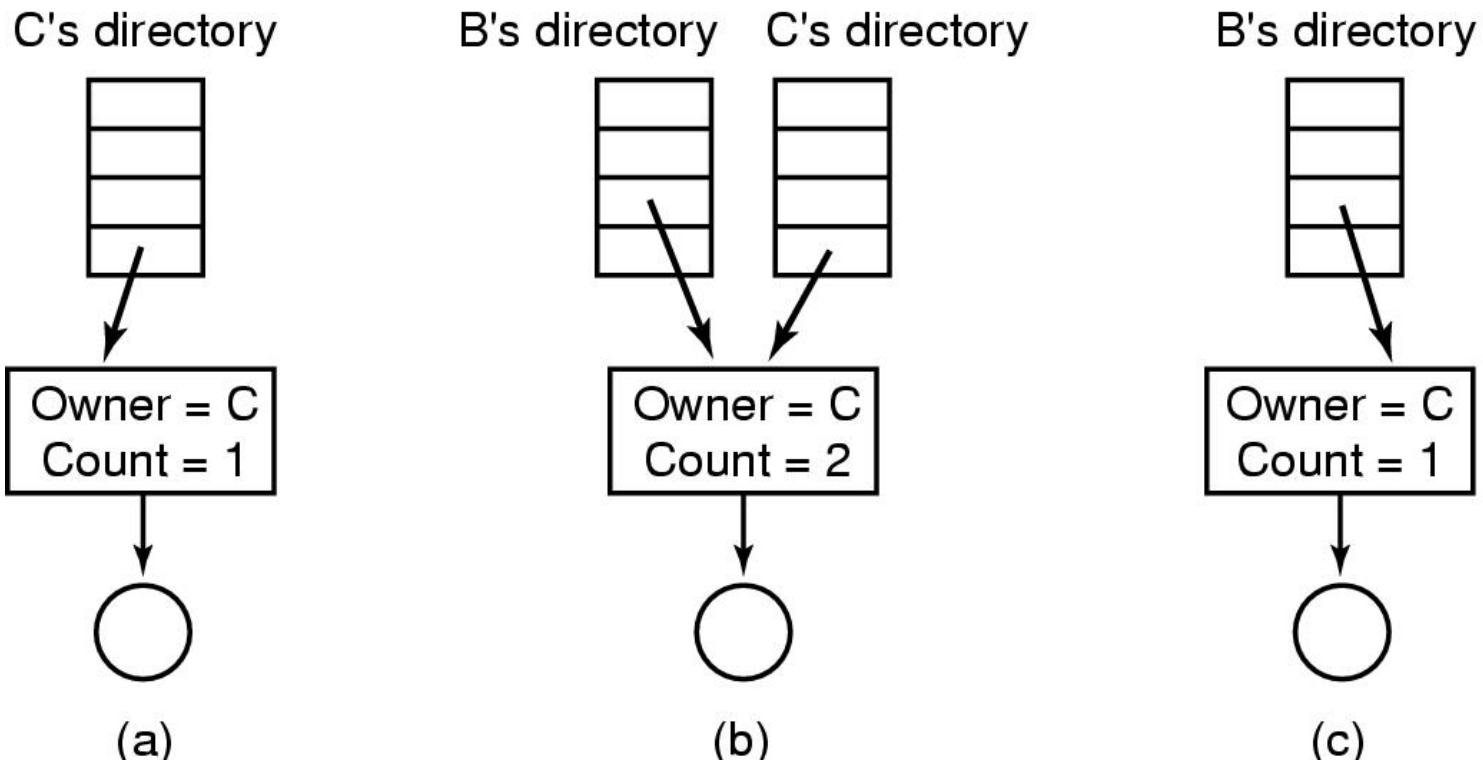
The steps in looking up */usr/ast/mbox*

Shared Files (1)



Hard and symbolic links

Hard Links

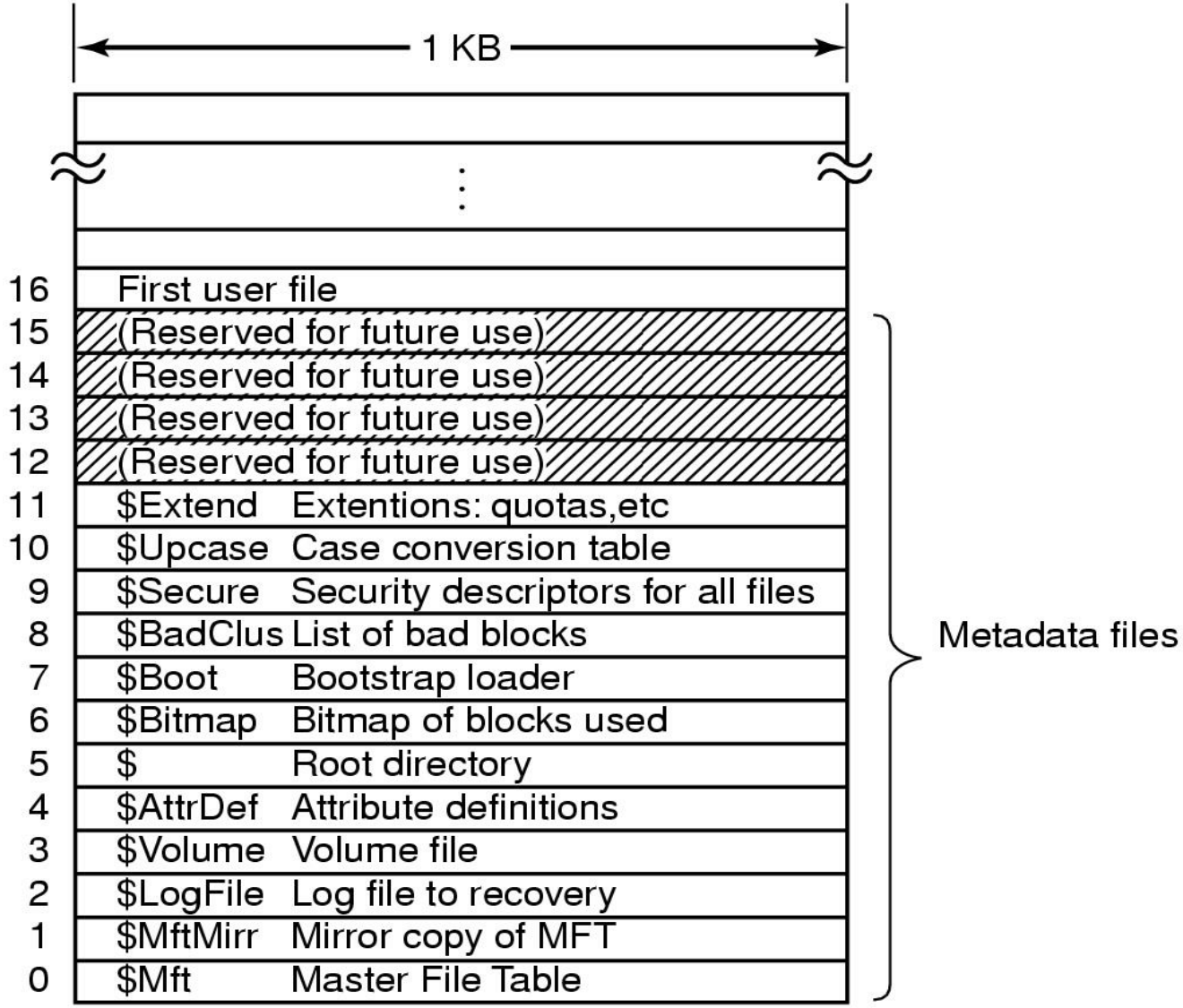


(a) Situation prior to linking

(b) After the link is created

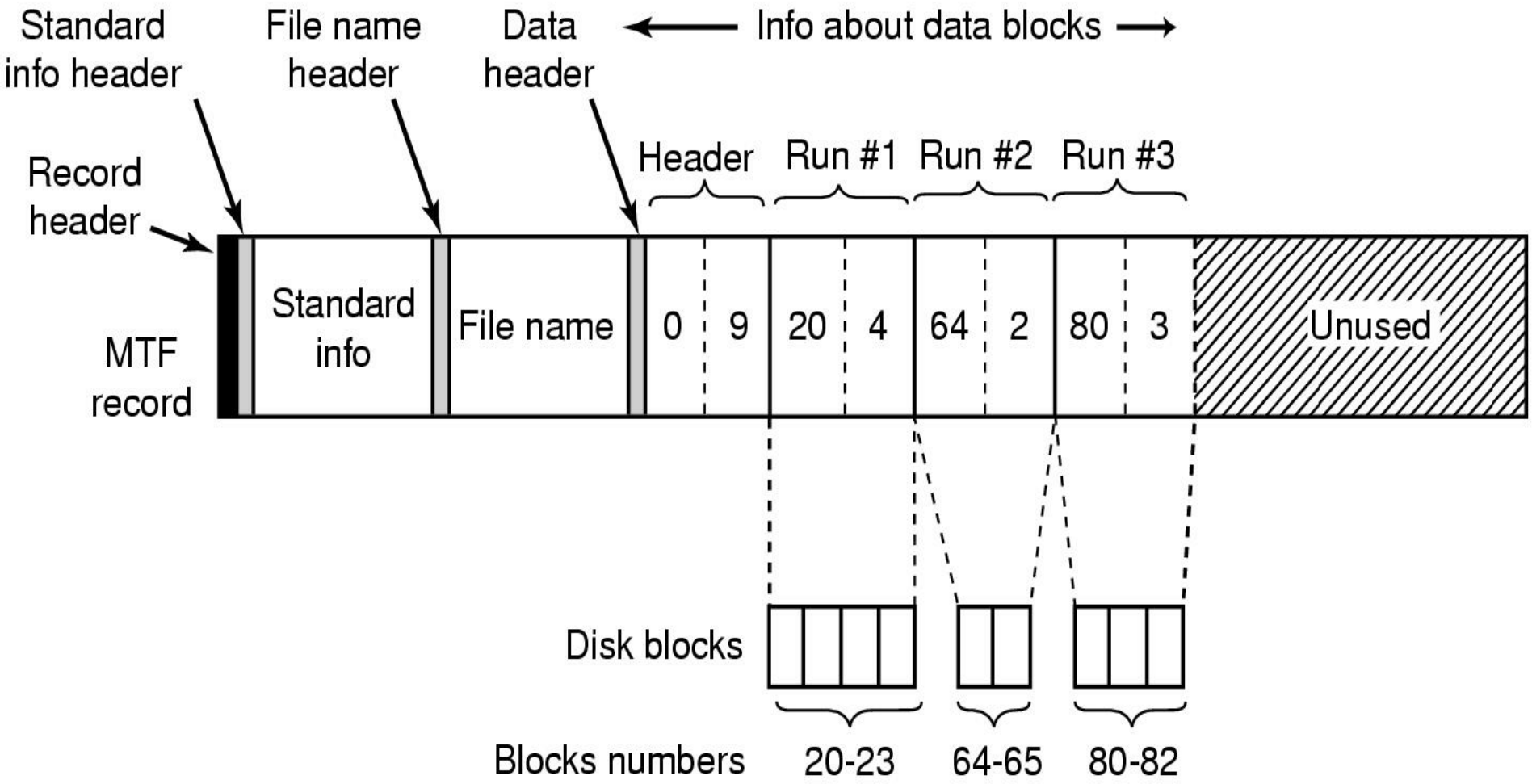
(c) After the original owner removes the file

Windows 2000 (1)



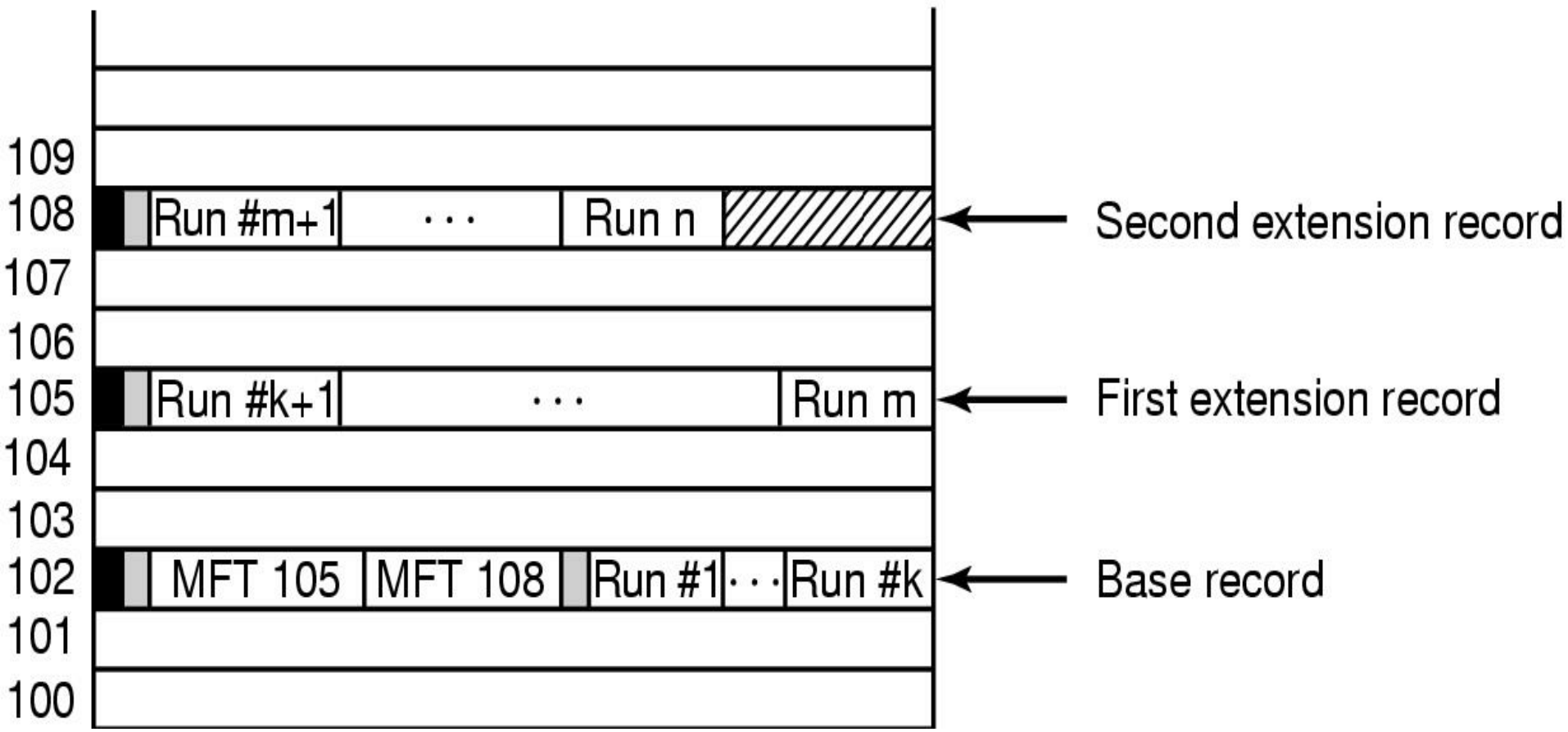
The NTFS master file table

Windows 2000 (2)



An MFT record for a three-run, nine-block file

Windows 2000 (3)



A file that requires three MFT records to store its runs