Chapter 12 - Slides

Stsallings 9ed

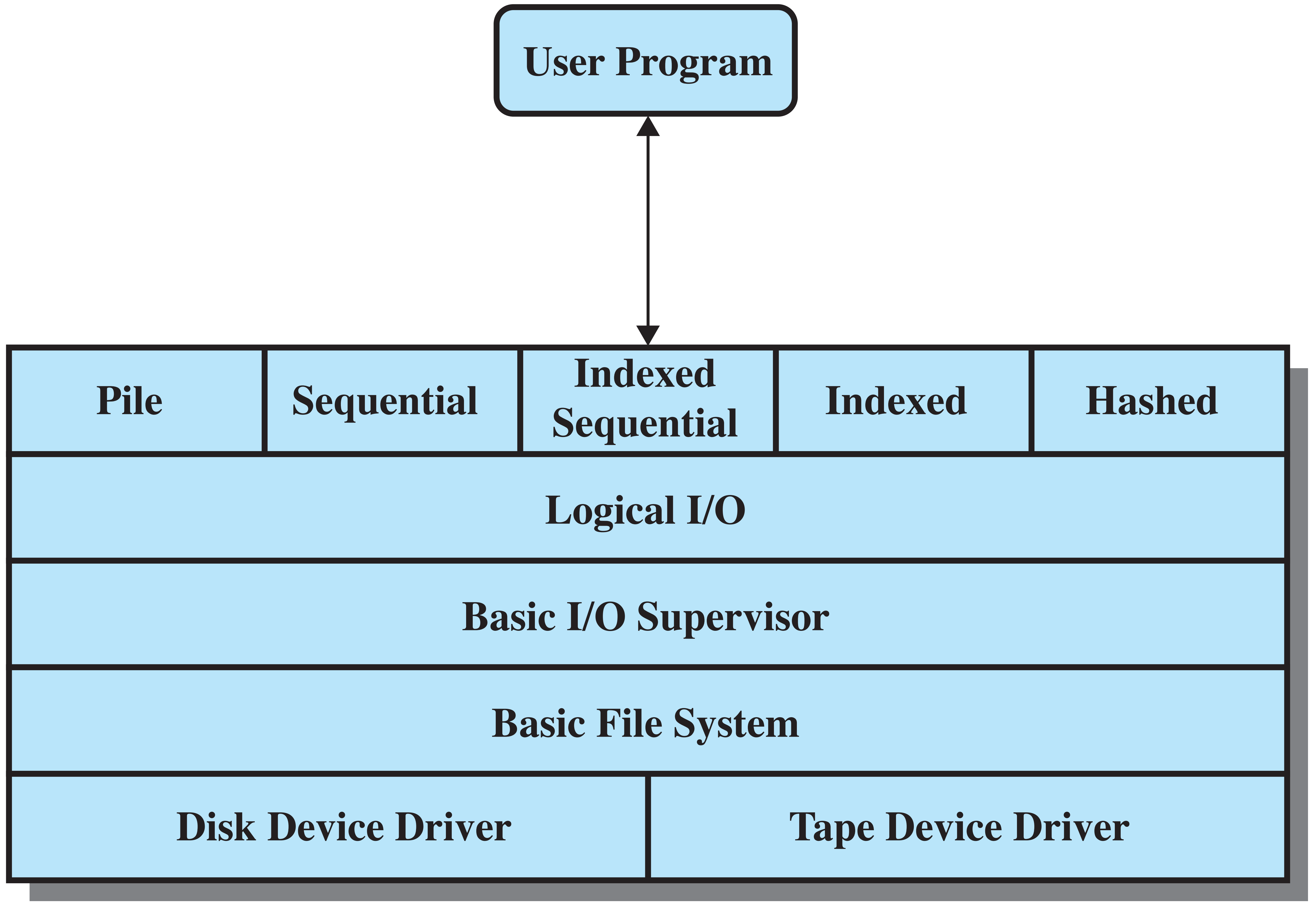


Figure 12.1 File System Software Architecture

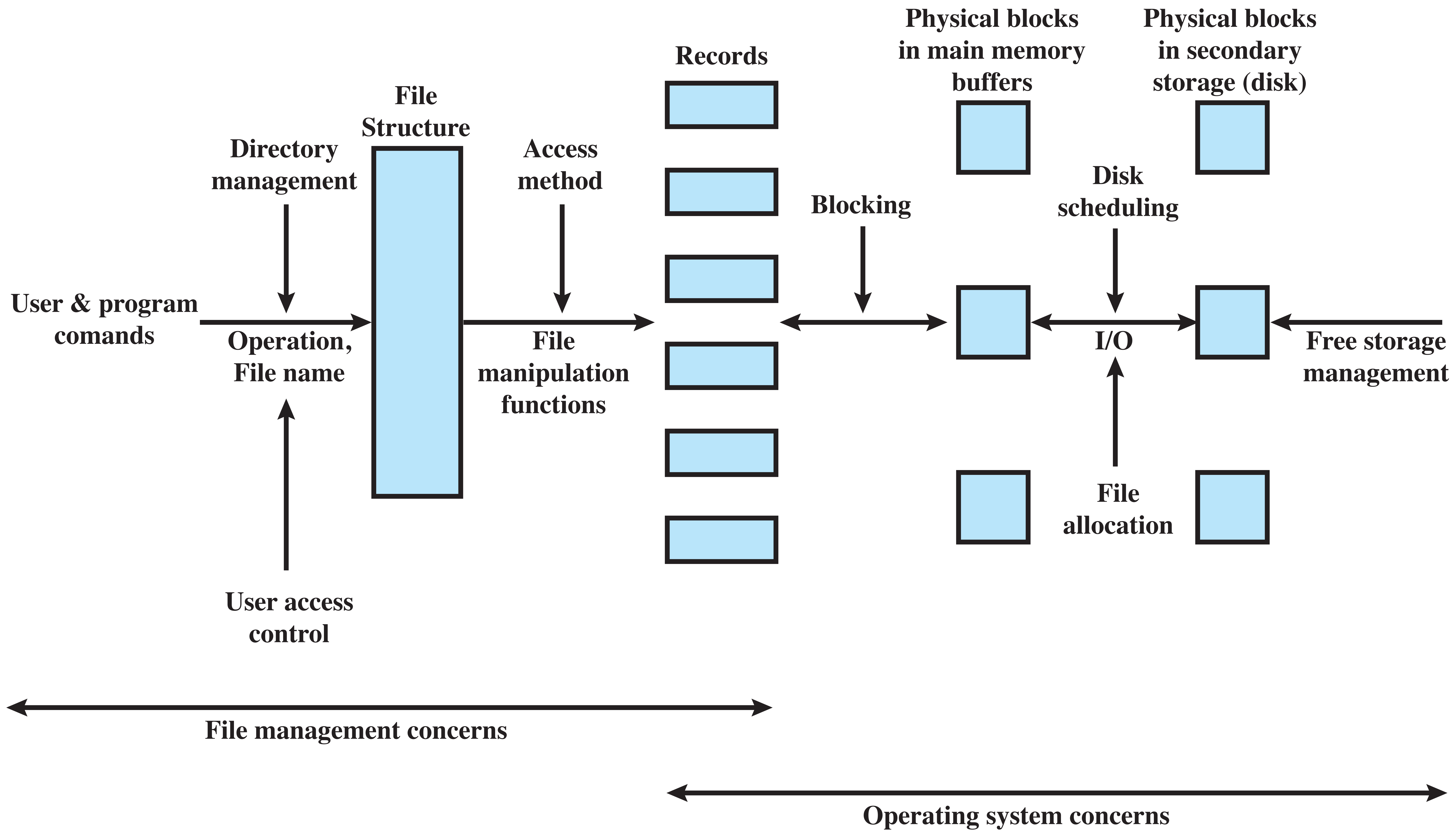
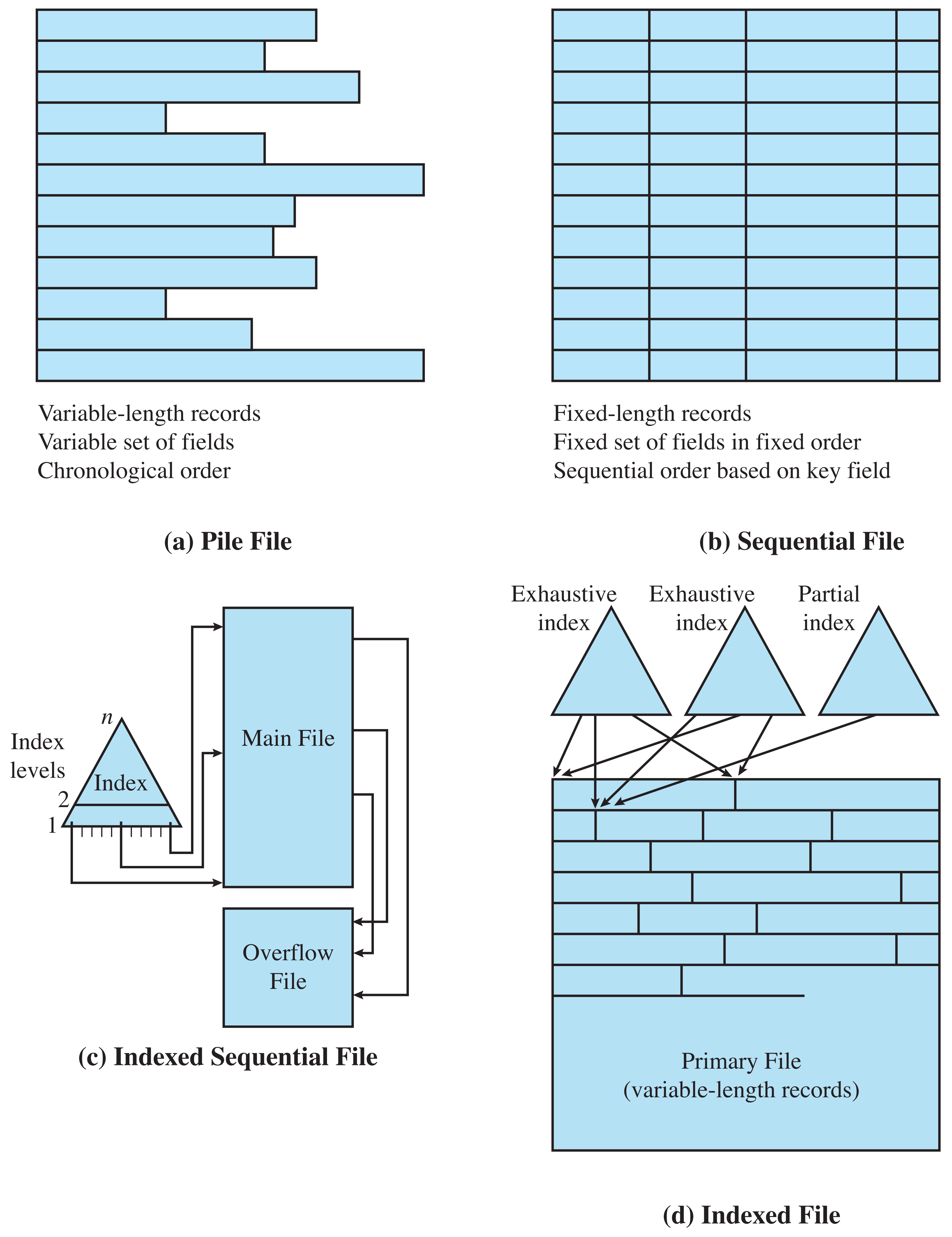


Figure 12.2 Elements of File Management



(a) Pile File

(b) Sequential File

(c) Indexed Sequential File

(d) Indexed File

Figure 12.3 Common File Organizations

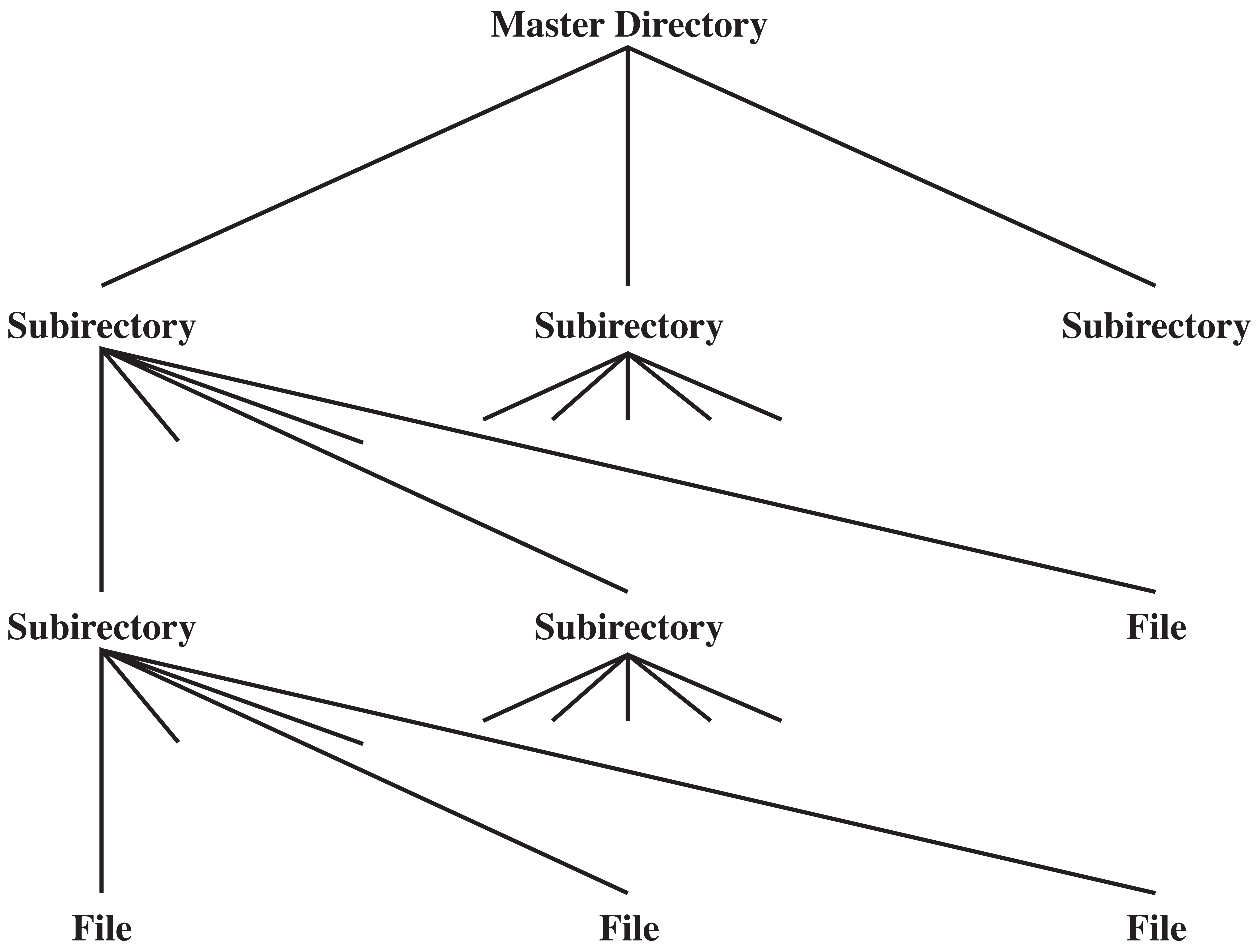


Figure 12.6 Tree-Structured Directory

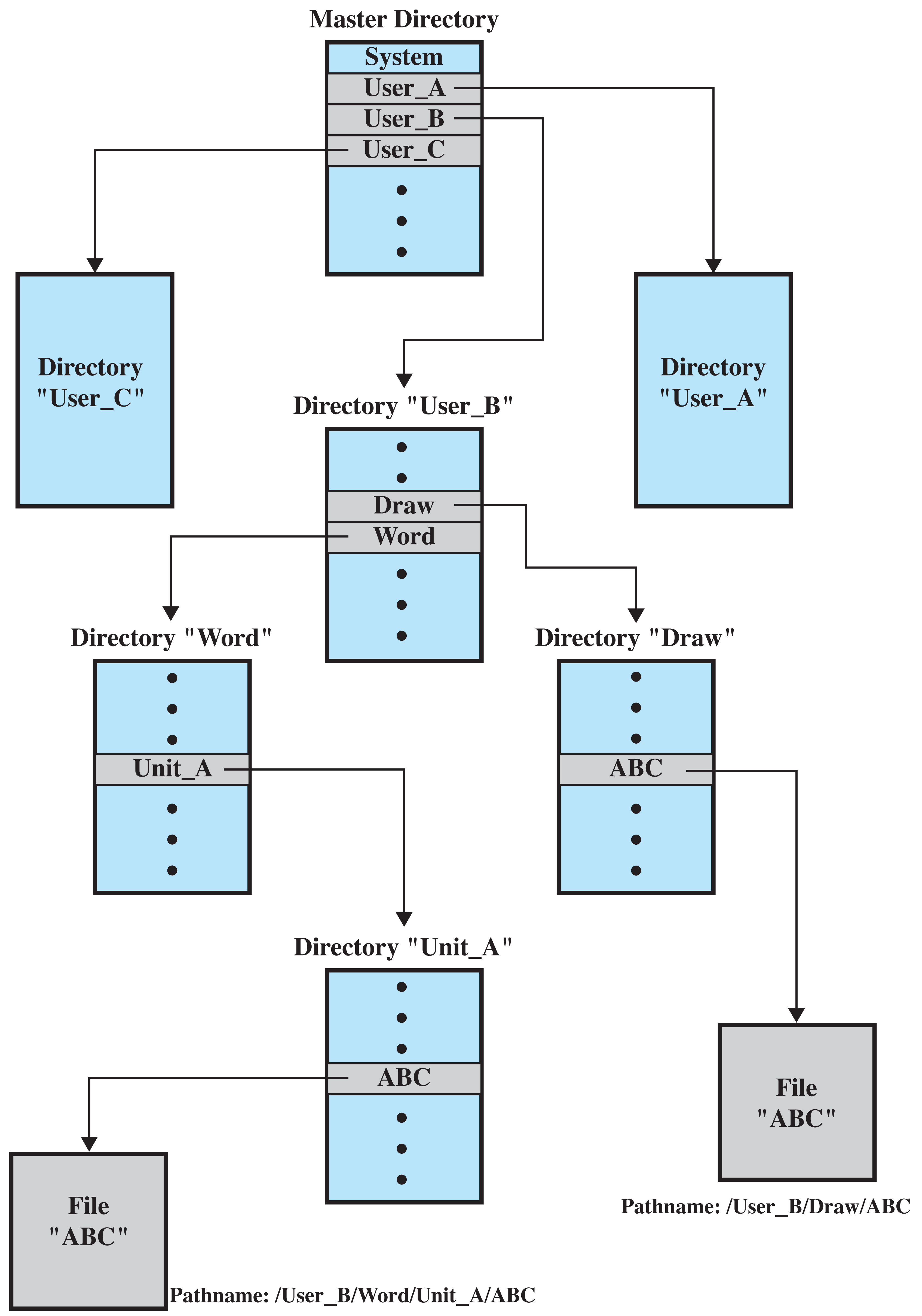
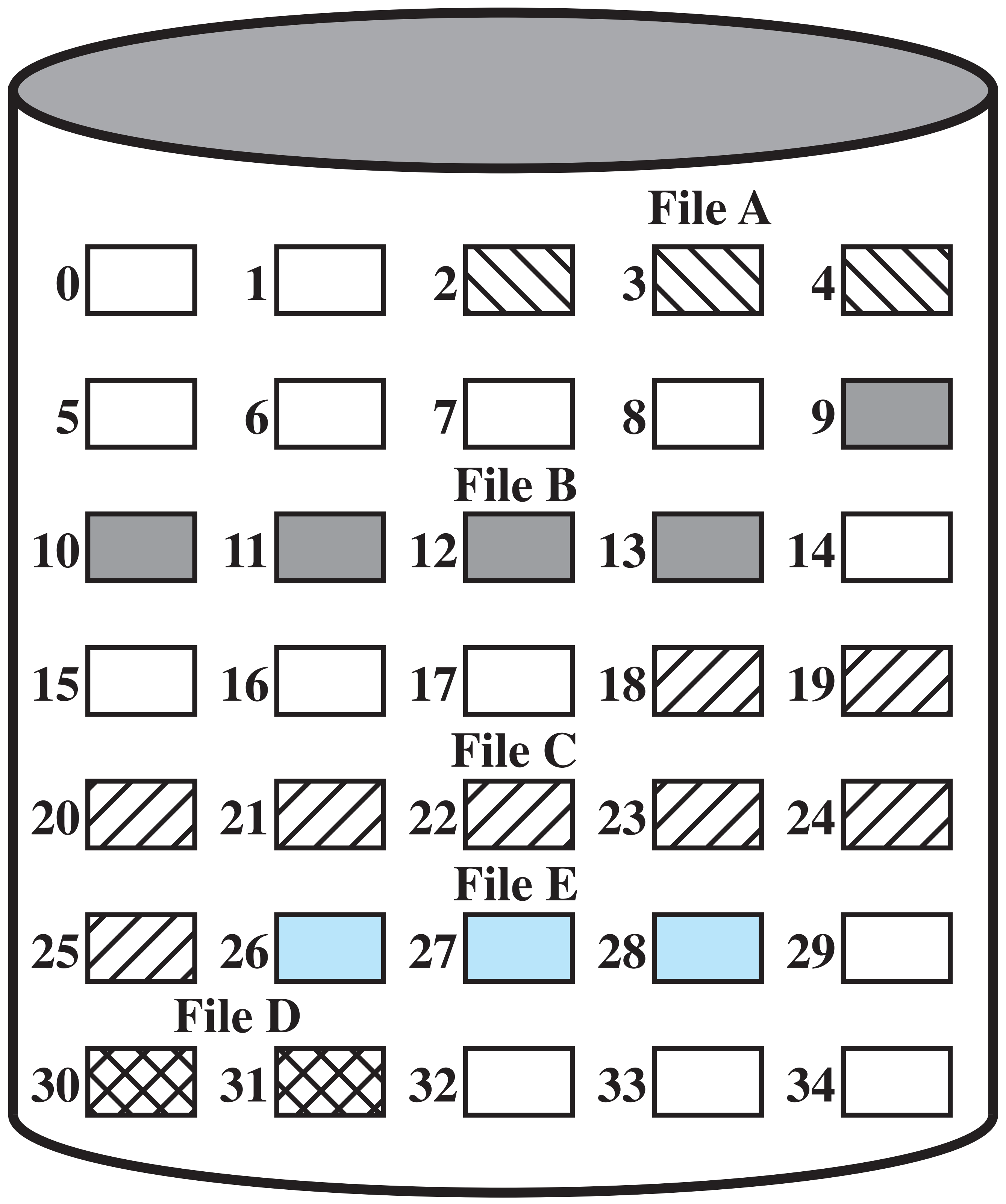


Figure 12.7 Example of Tree-Structured Directory

Disk Allocation Methods

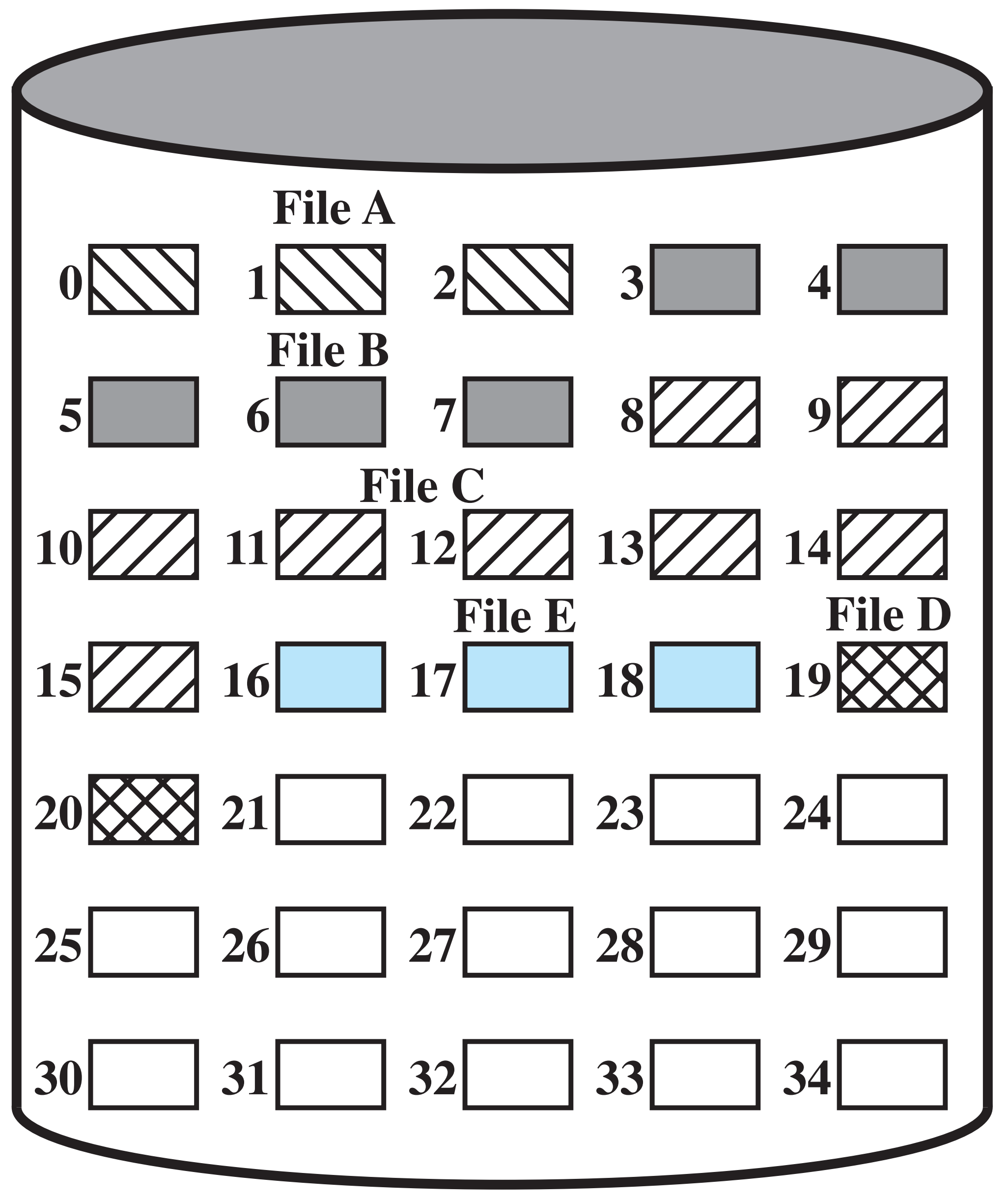
- Contiguous
  - Simple and efficient, but
  - Not very flexible, but
  - Other formats strive for it
- Linked List
  - Great for sequential access, but
  - Not so good for random access.
  - File Allocation Table (FAT) – links in separate table
- Indexed
  - Good for both random and sequential access, but
  - Large files require lots of indices.
  - Unix uses indirect blocks



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

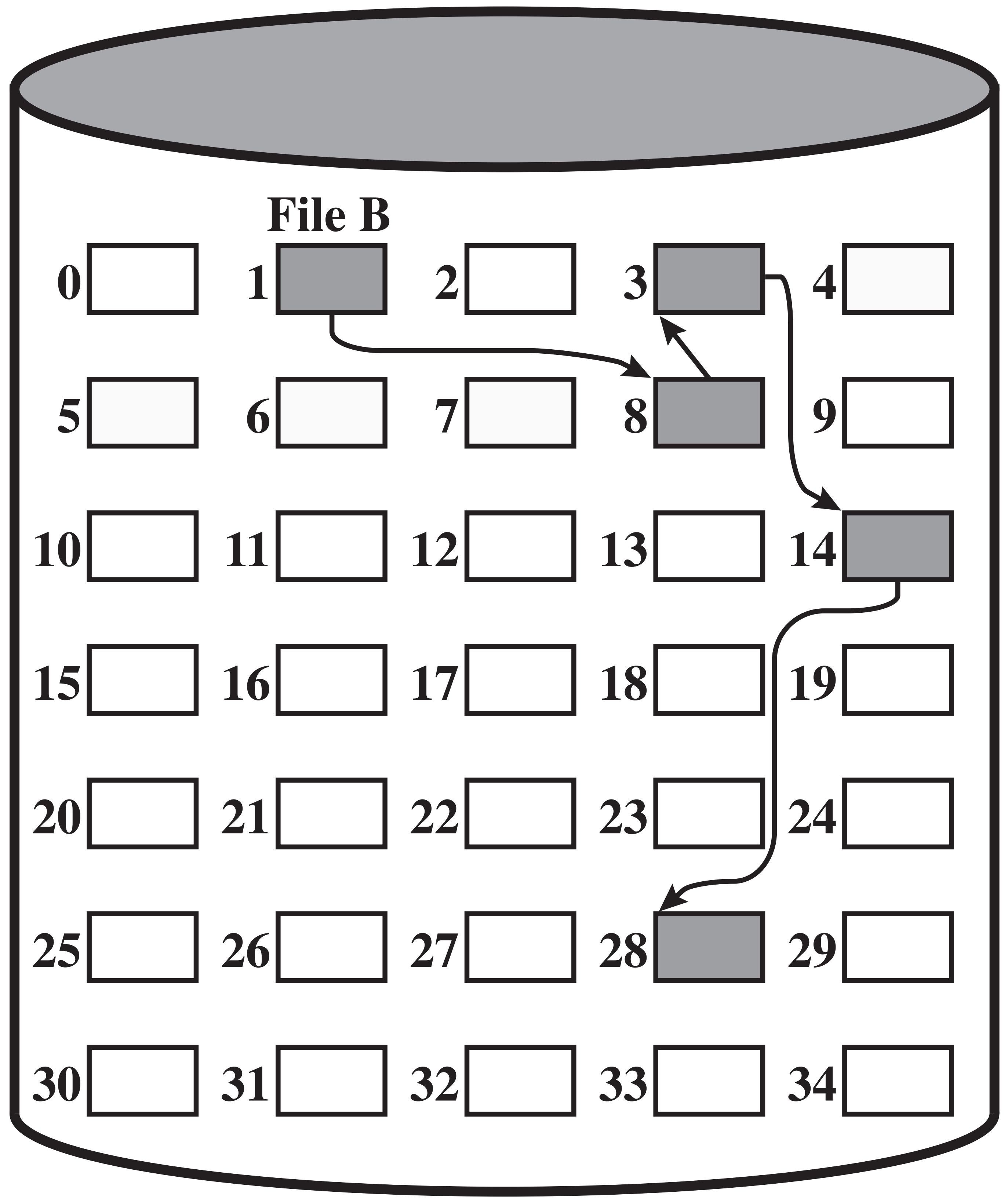
Figure 12.9 Contiguous File Allocation



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

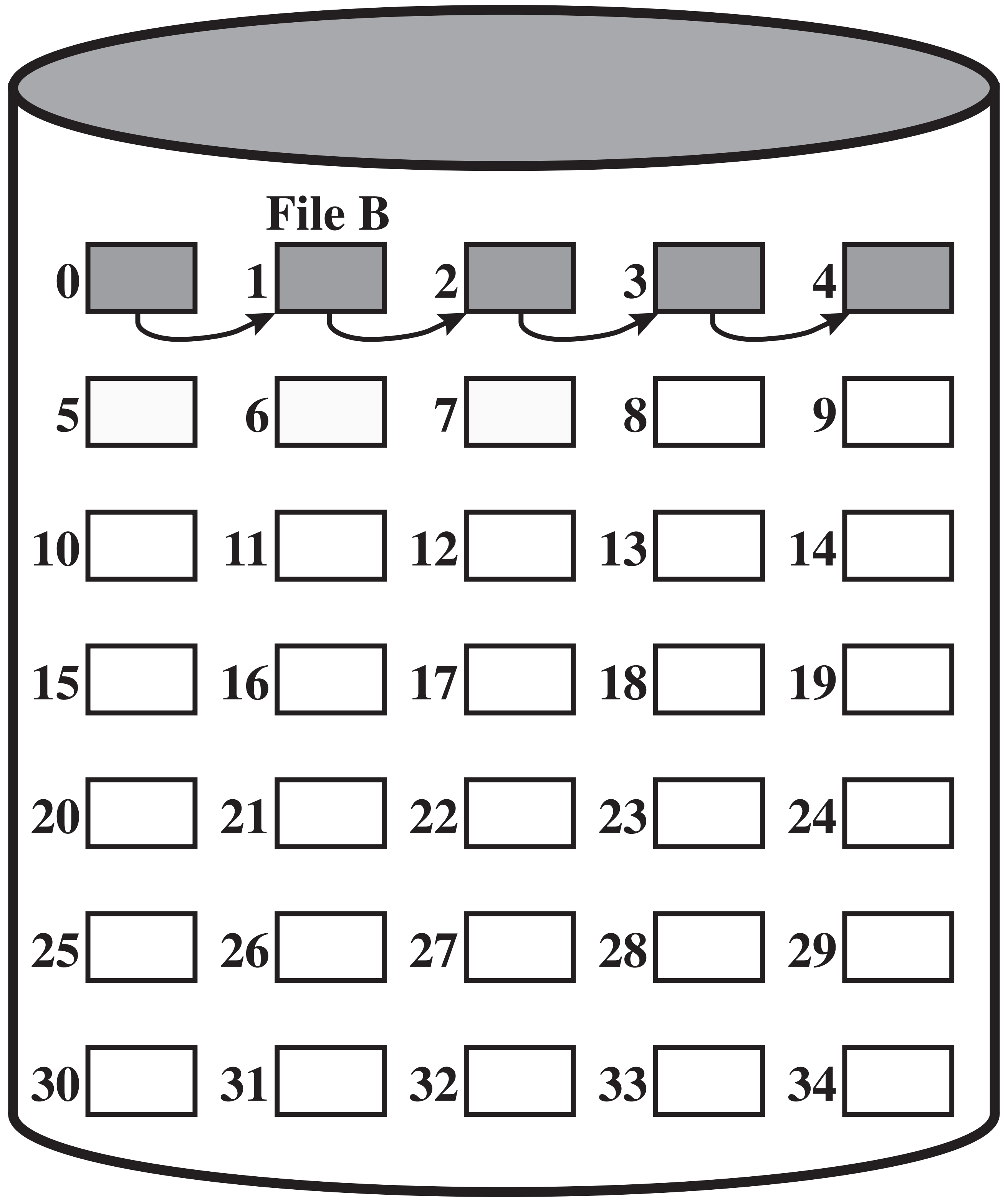
Figure 12.10 Contiguous File Allocation (After Compaction)



File Allocation Table

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

Figure 12.11 Chained Allocation

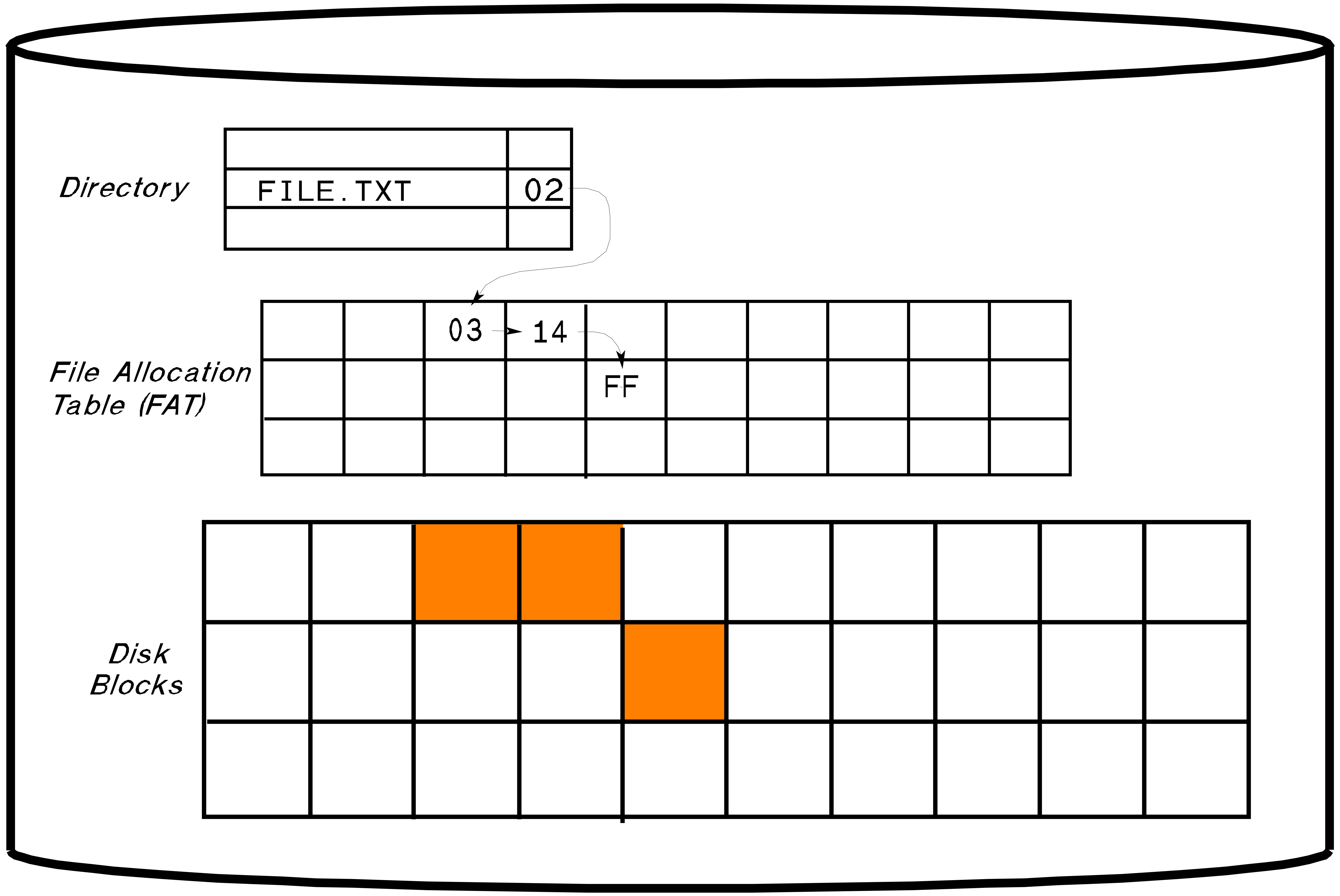


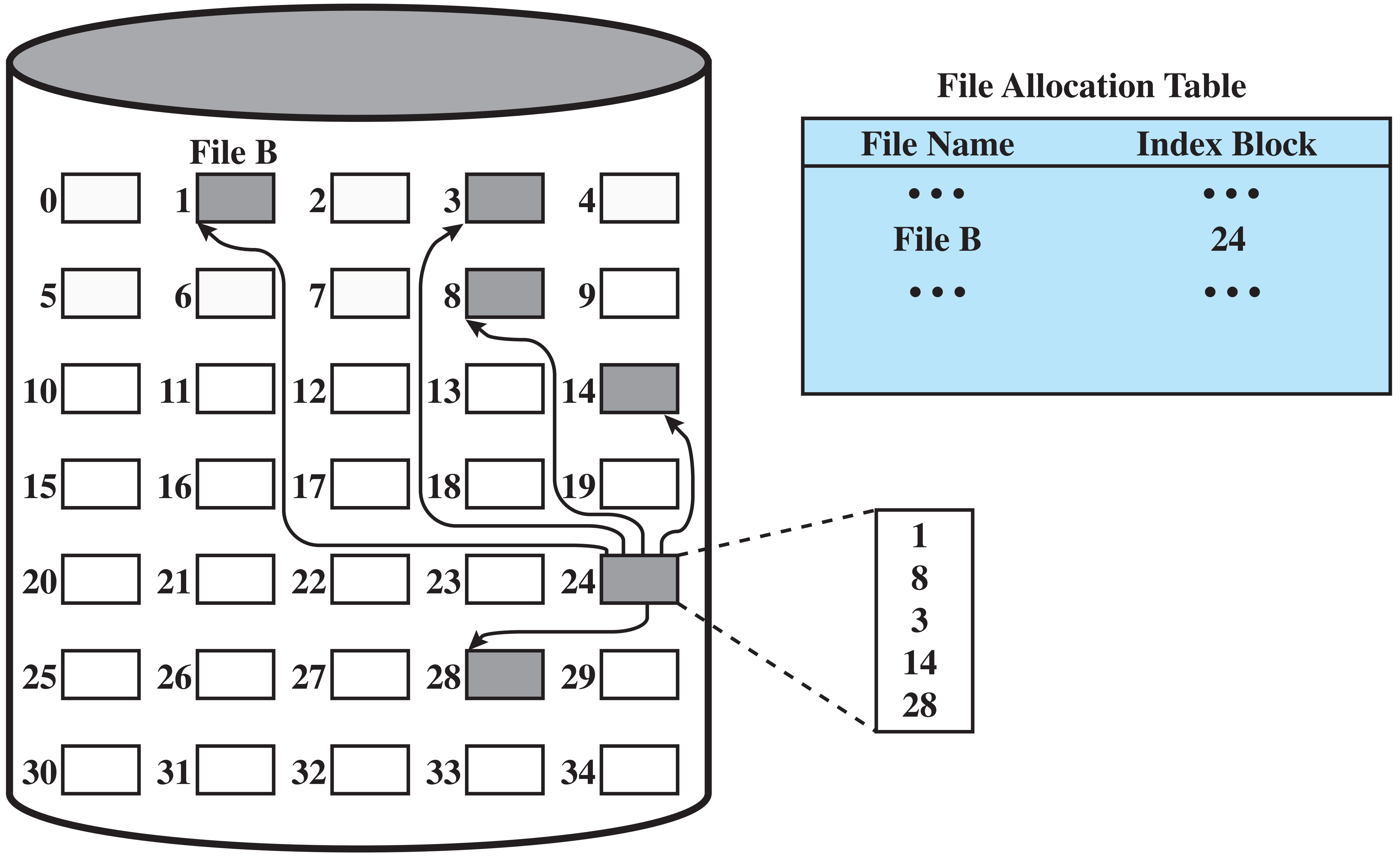
File Allocation Table

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

Figure 12.12 Chained Allocation (After Consolidation)

FAT File Example

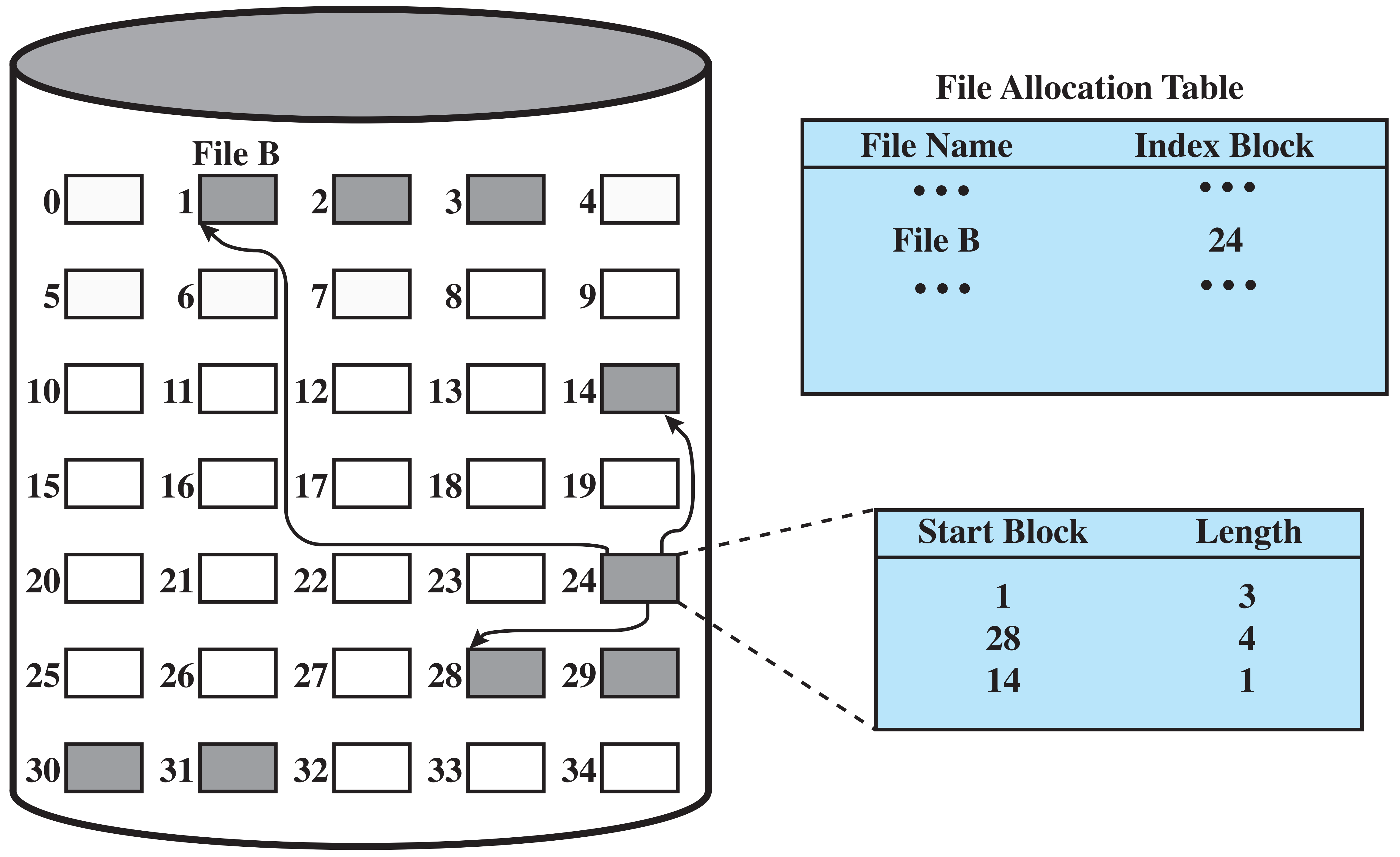




File Allocation Table

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

Figure 12.13 Indexed Allocation with Block Portions



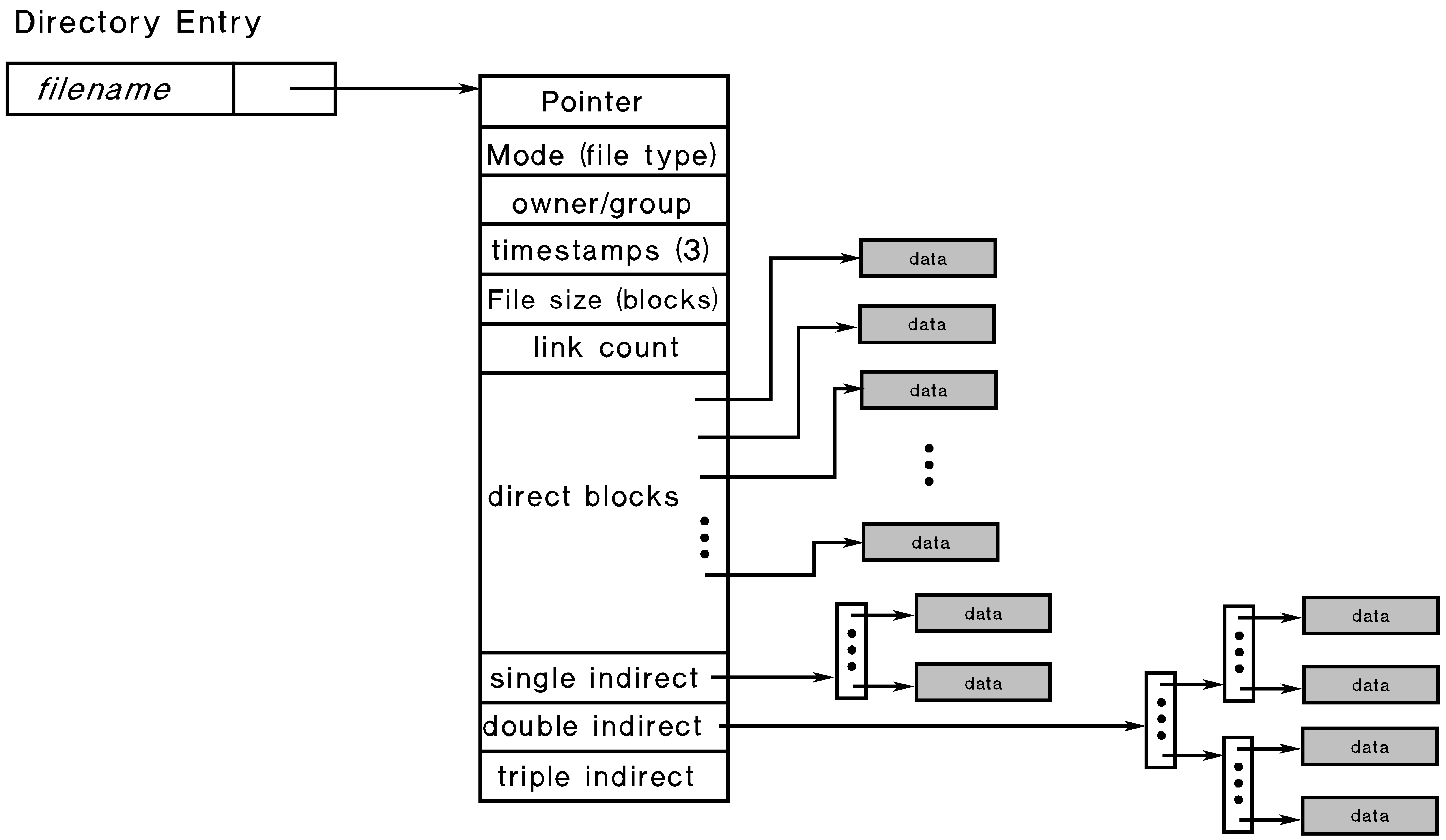
File Allocation Table

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

Start Block	Length
1	3
28	4
14	1

Figure 12.14 Indexed Allocation with Variable-Length Portions

UNIX i-node


Directory Entry
filename
Pointer
Mode (file type)
owner/group
timestamps (3)
File size (blocks)
link count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

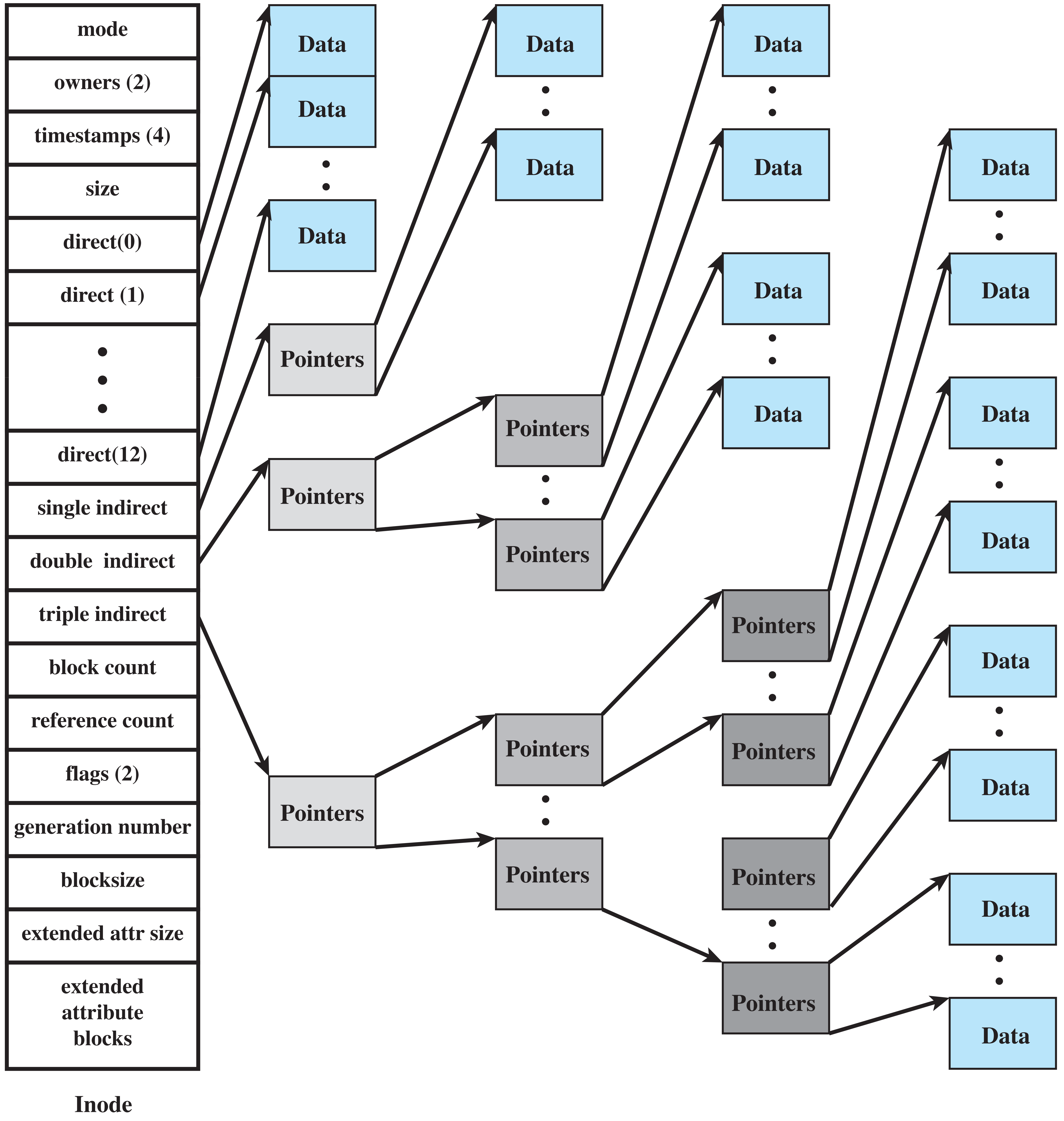


Figure 12.15 Structure of FreeBSD inode and File

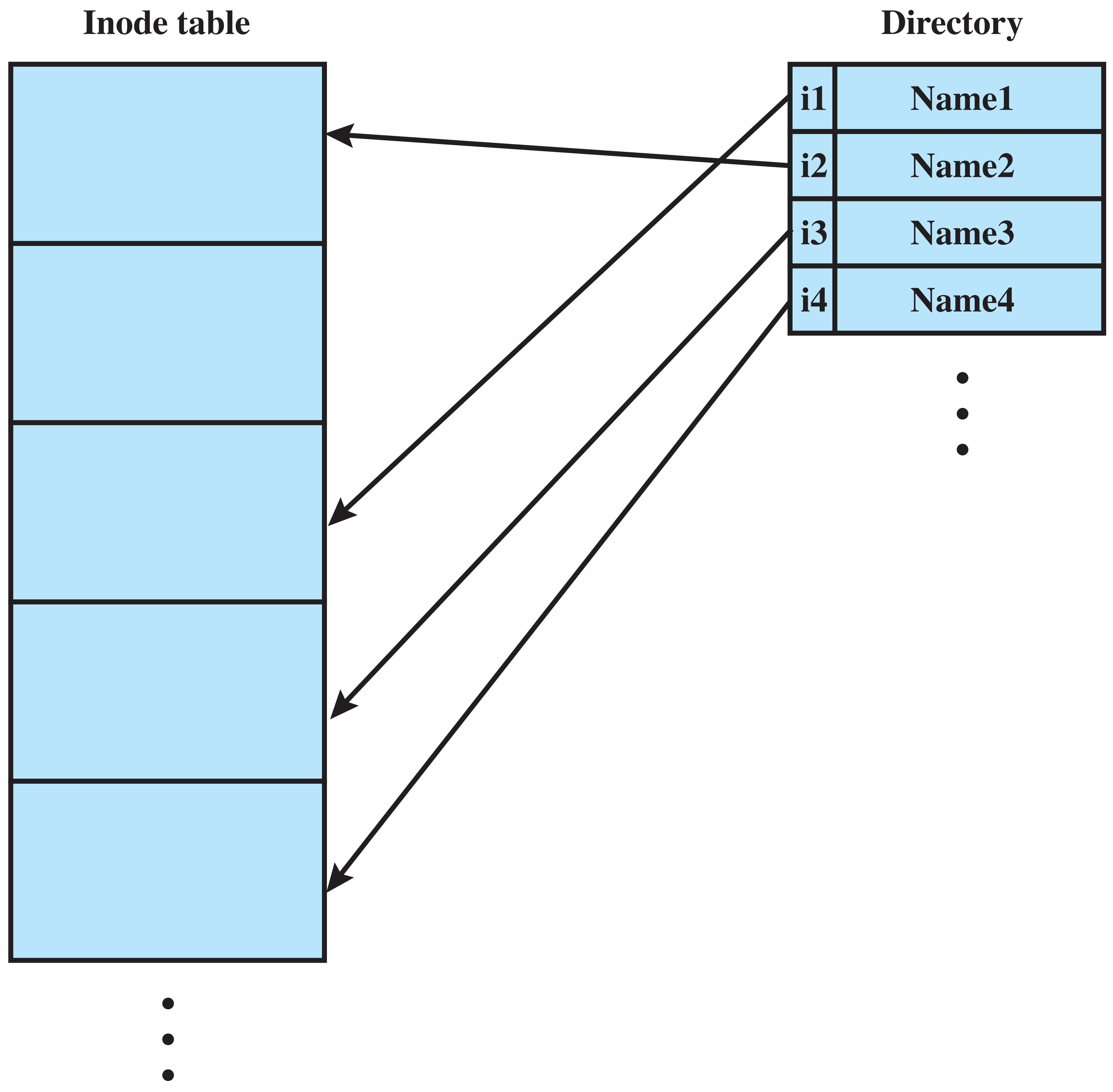


Figure 12.16 UNIX Directories and Inodes

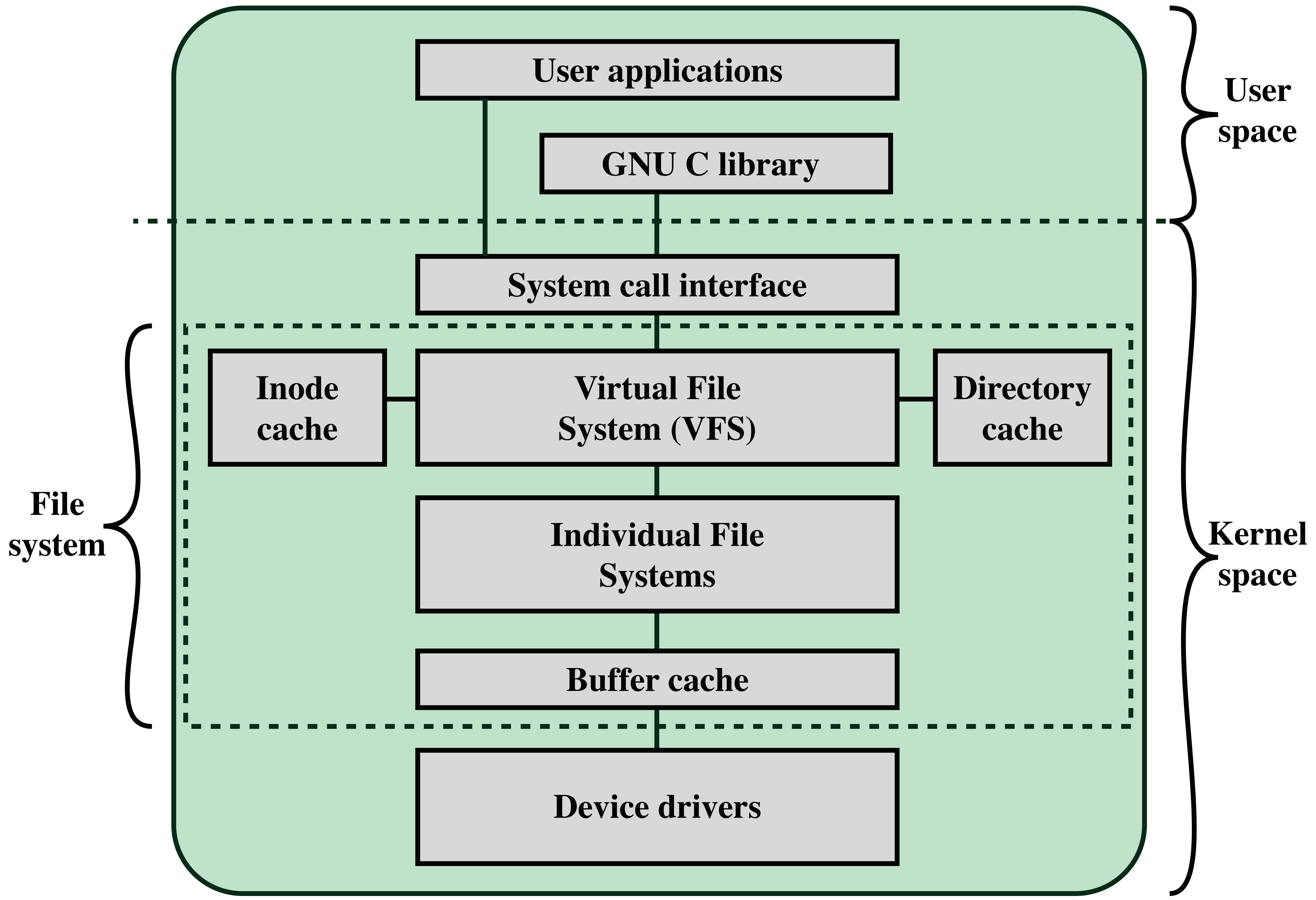


Figure 12.17 Linux Virtual File System Context

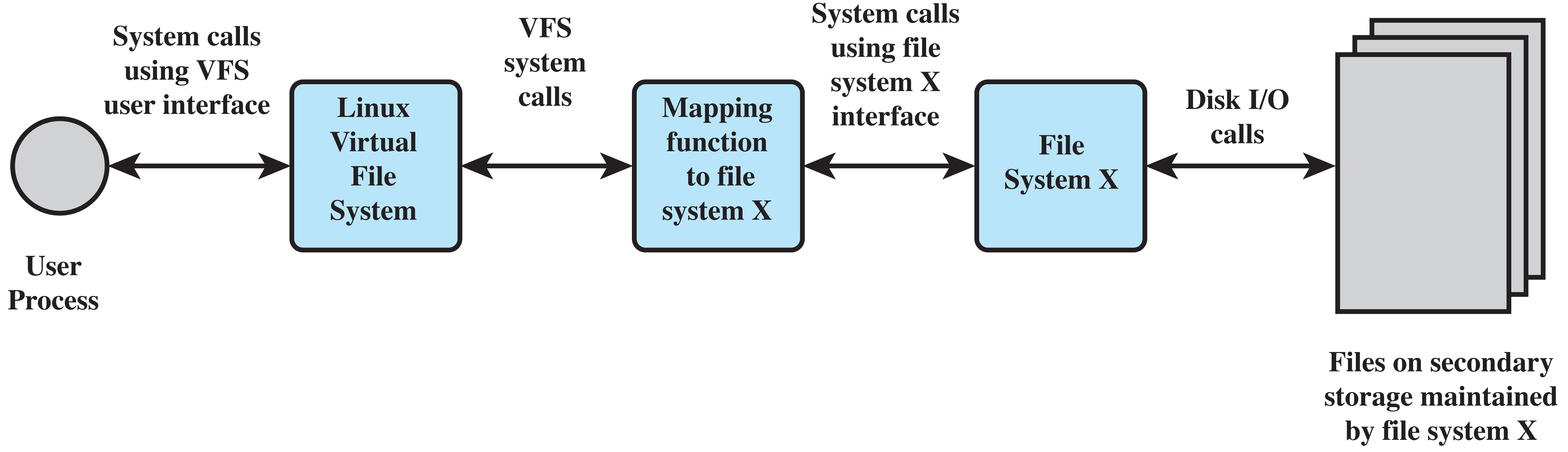

System calls using VFS user interface
User Process
Linux Virtual File System
VFS system calls
Mapping function to file system X
System calls using file system X interface
File System X
Disk I/O calls
Files on secondary storage maintained by file system X

Figure 12.18 Linux Virtual File System Concept

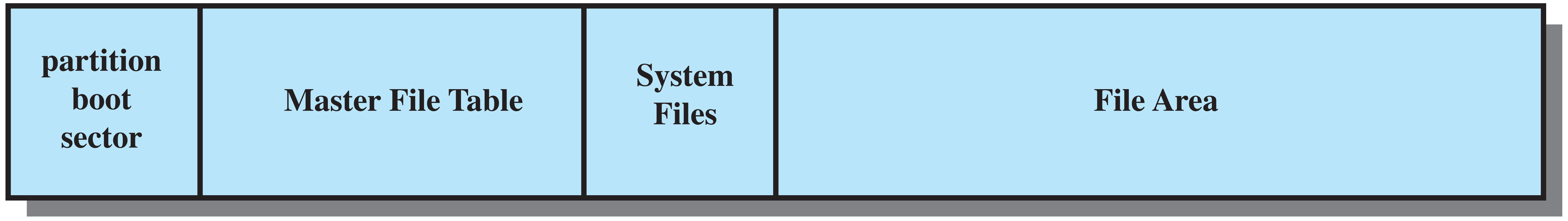
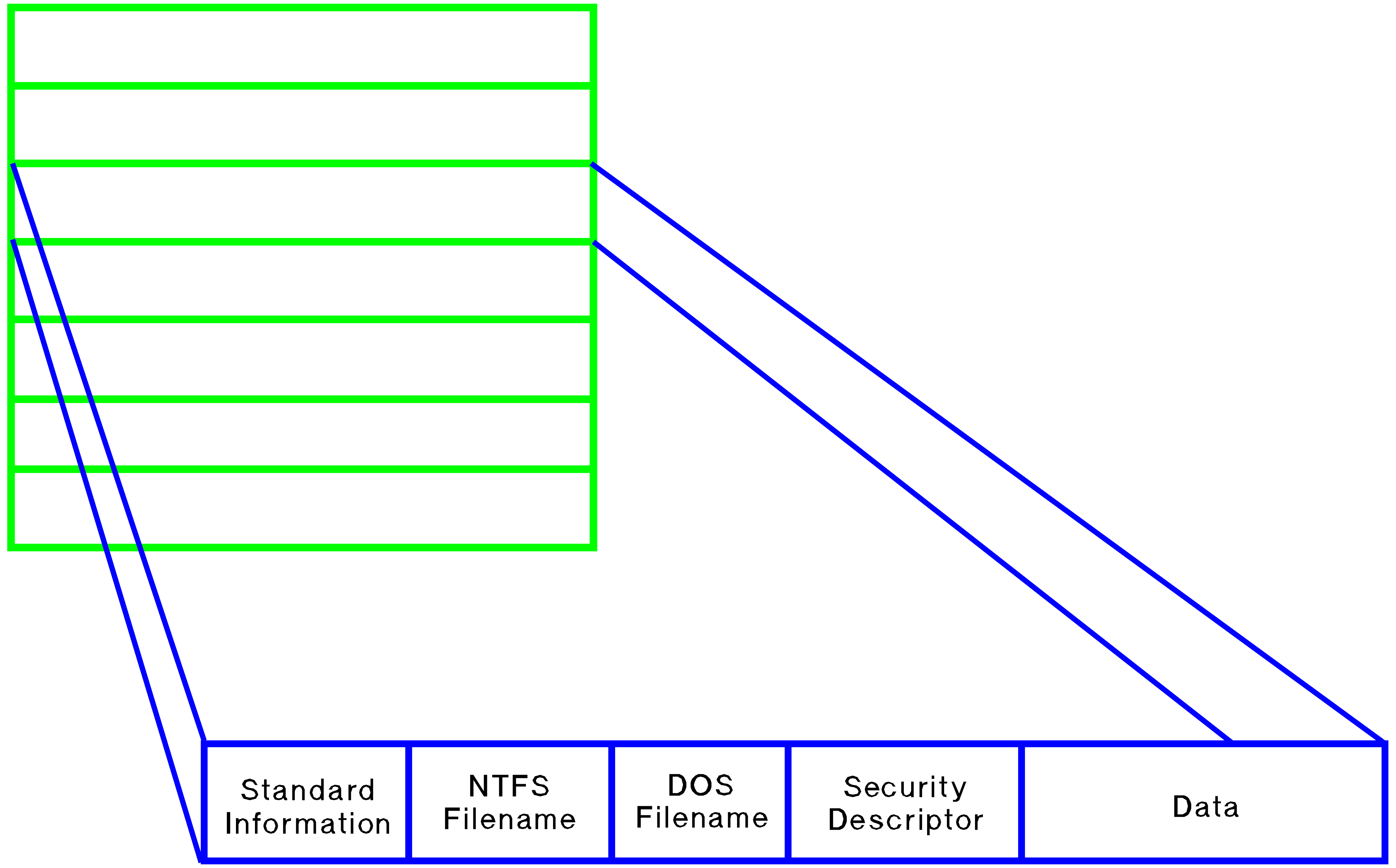


Figure 12.19 NTFS Volume Layout

NTFS Metadata Files

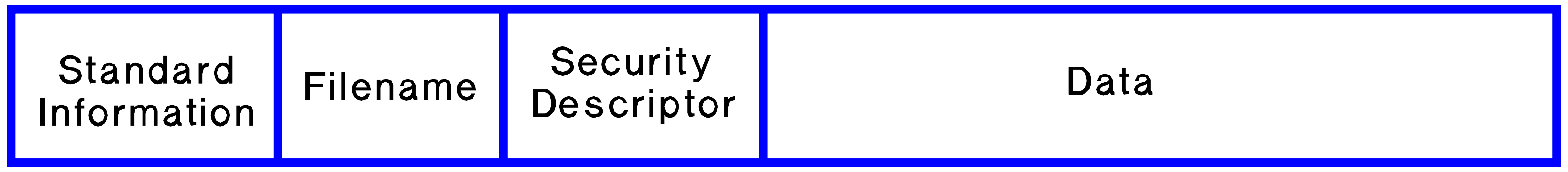
File 0	MFT ($MFT)
1	MFT copy ($MFTmirr)
2	Log file ($LogFile)
3	Volume file ($Volume)
4	Attribute defn table ($Attrdef)
5	Root Directory (\\)
6	Bitmap file ($Bitmap)
7	Boot file ($Boot)
8	Bad-Cluster file ($BadClus)
	⋮
	User files/directories

NTFS Master File Table

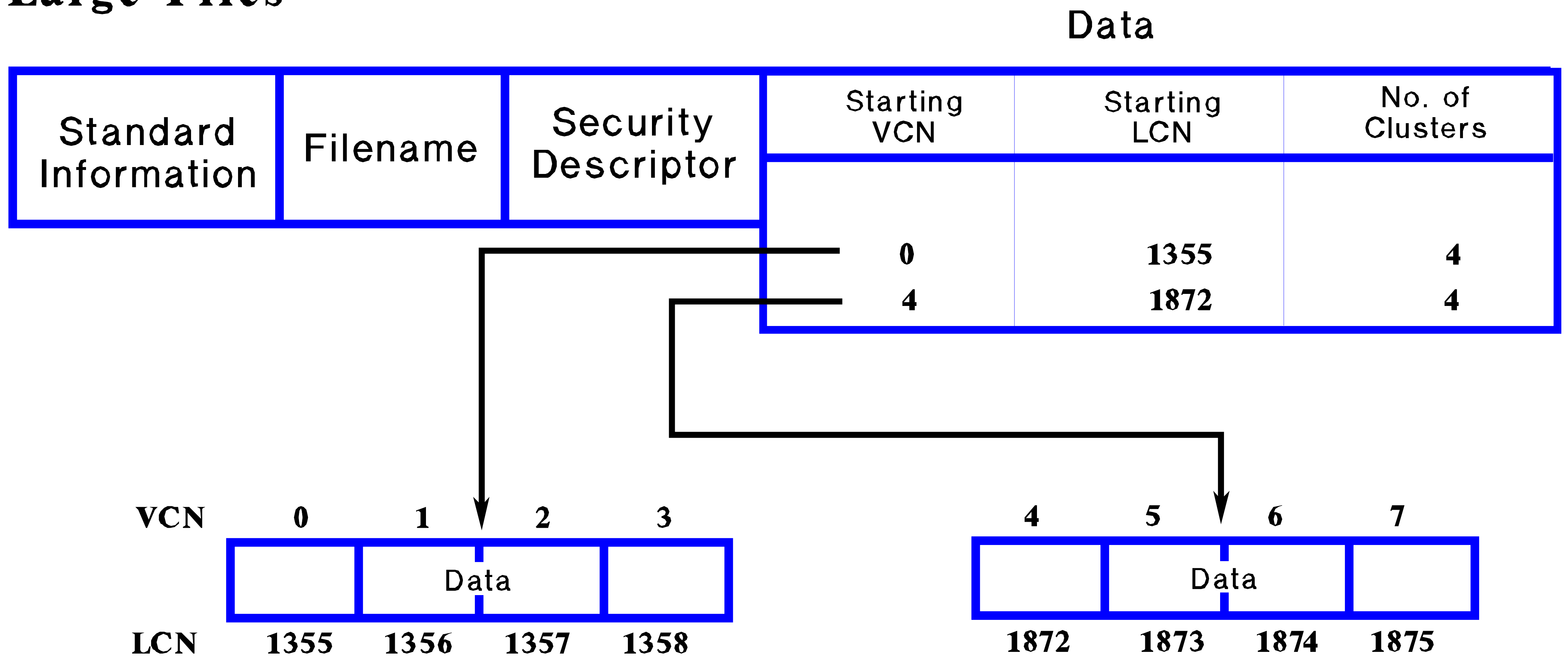


NTFS "Regular" Files

Small Files



Large Files

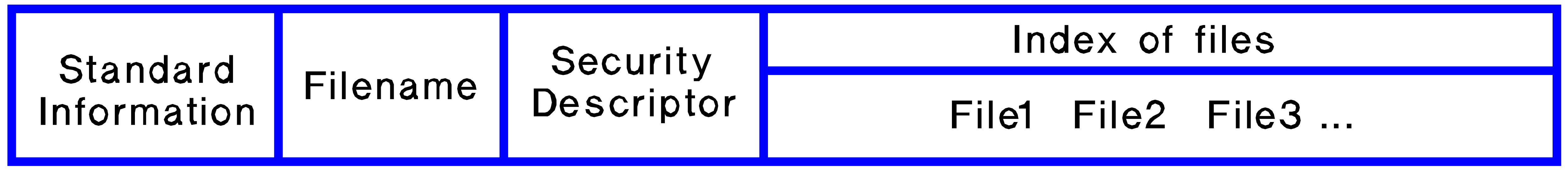


NTFS0030

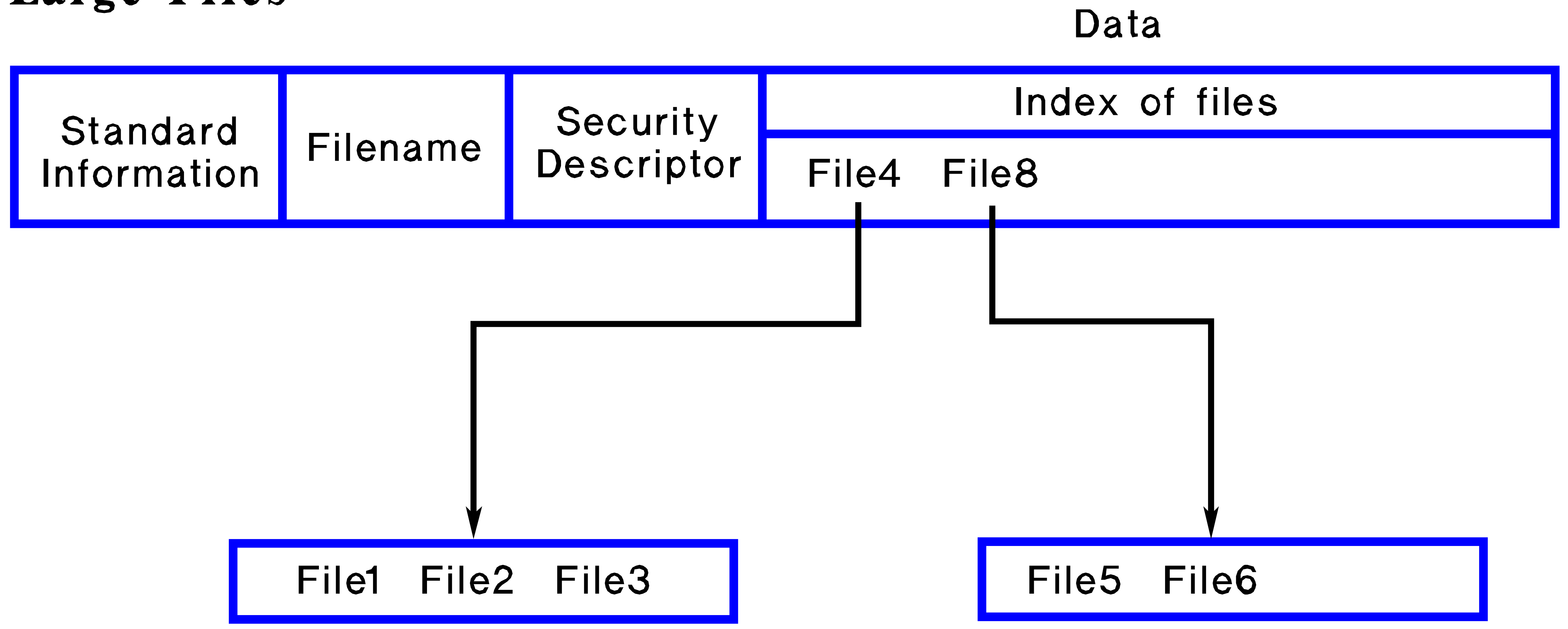
University of Idaho

NTFS Directory Files

Small Files



Large Files



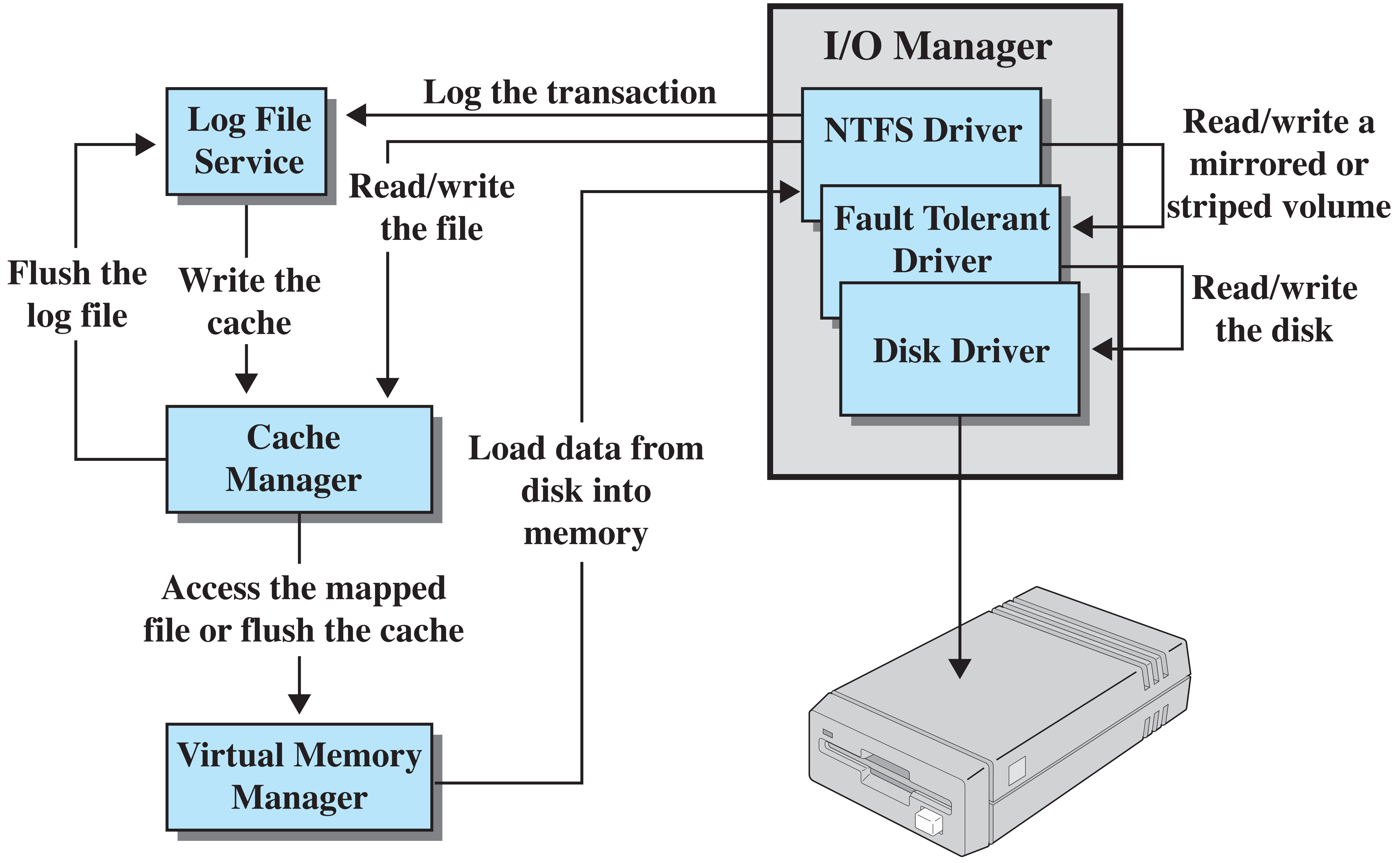
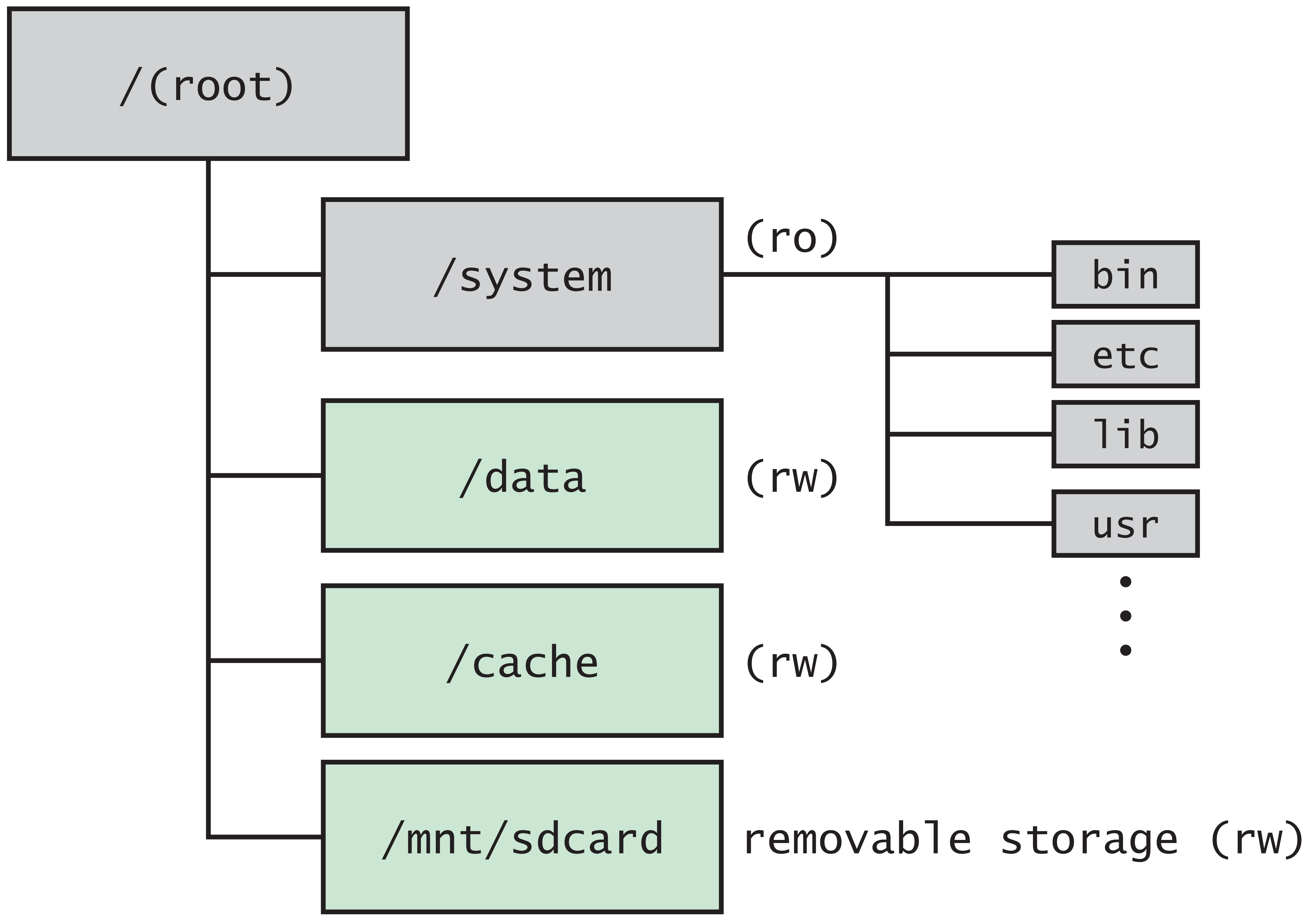


Figure 12.20 Windows NTFS Components



ro: mounted as read only
rw: mounted as read and write

Figure 12.21 Typical Directory Tree of Android