File Management

Chapter 12

Contents

- Overview
- File organization and Access
- File directories
- File sharing
- Record blocking
- Secondary storage management
- Unix file management
- Windows 2000 file system

File Management

- In most applications, the file is the central element
  - Input to applications is by means of a file
  - Output is saved in a file for long-term storage
- File management system is considered part of the operating system

Terms Used with Files

- Field
  - basic element of data
  - contains a single value
  - characterized by its length and data type
- Record
  - collection of related fields
  - treated as a unit
    - Example: employee record

Terms Used with Files

- File
  - collection of similar records
  - treated as a single entity by users
  - have unique names
  - access control apply at this level
    - in some systems, such controls are enforced at the record level
- Database
  - collection of related data
  - consists of one or more types of files

Typical Operations on Files

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

File Management System

- Set of system software that provides services to users and applications in the use of files
  - the only way that a user or application may access files is through the file management system

Objectives for a File Management System

- Meet the data management needs and requirements of the user
- Guarantee that the data in the file are valid
- Optimize performance
  - throughput and response time
- Provide I/O support for a variety of storage device types

Objectives for a File Management System

- 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

- Each user should be able to create, delete, read, and change files
- Each user may have controlled access to other users' files
- Each user may control what type of accesses are allowed to the user's files
- Each user should be able to restructure the user's files in a form appropriate to the problem

Minimal Set of Requirements

- Each user should be able to move data between files
- Each user should be able to back up and recover the user’s files in case of damage
- Each user should be able to access the user’s files by using symbolic names

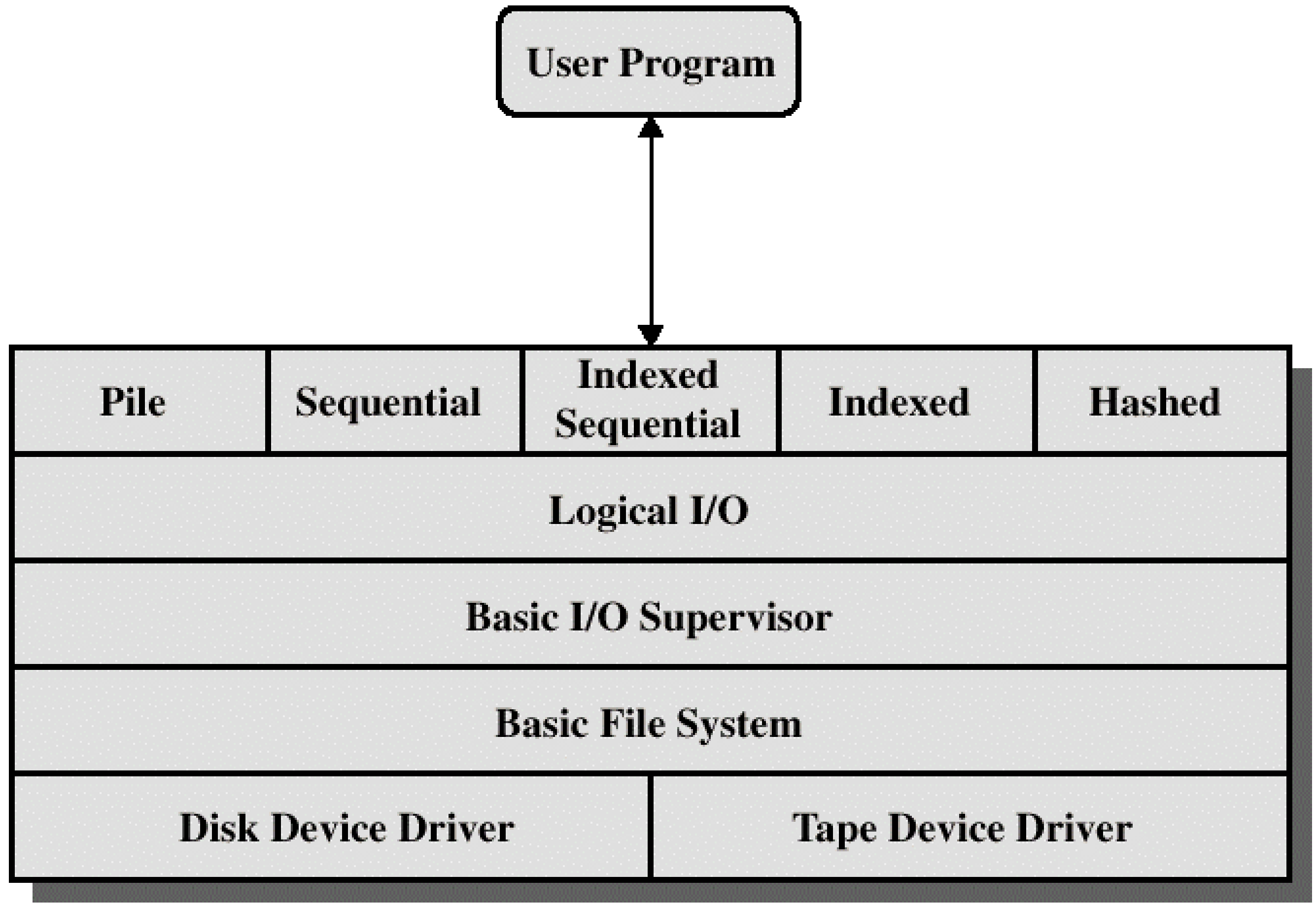


Figure 12.1 File System Software Architecture [GROS86]

Criteria for File Organization

- Rapid access
- Ease of update
- Economy of storage
- Simple maintenance
- Reliability

Criteria for File Organization

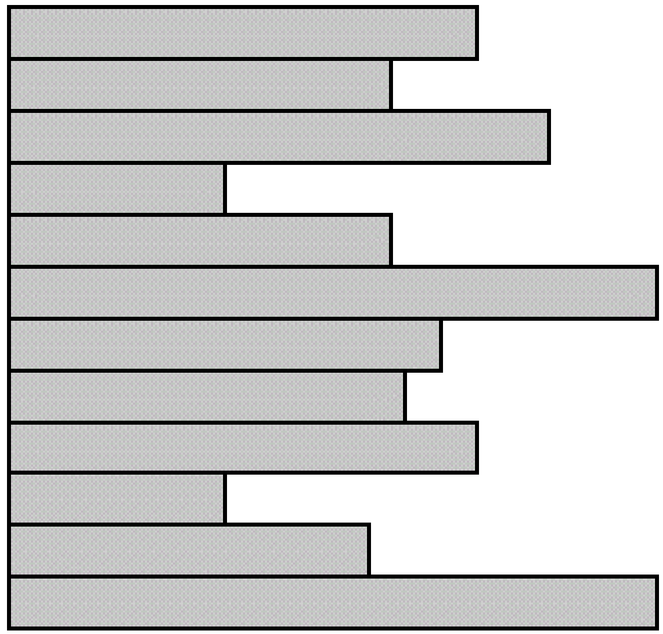
- These criteria may conflict
  - for economy of storage, there should be minimum redundancy in the data
  - redundancy is the primary means of increasing the speed of access to data
    - index

File Organization

- Pile
- Sequential file
- Indexed sequential file
- Indexed file
- Direct, or hashed, file

File Organization

- 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
    - length of a record must be specified either implicitly or explicitly
  - no structure
  - record access is by exhaustive search



Variable-length records
Variable set of fields
Chronological order

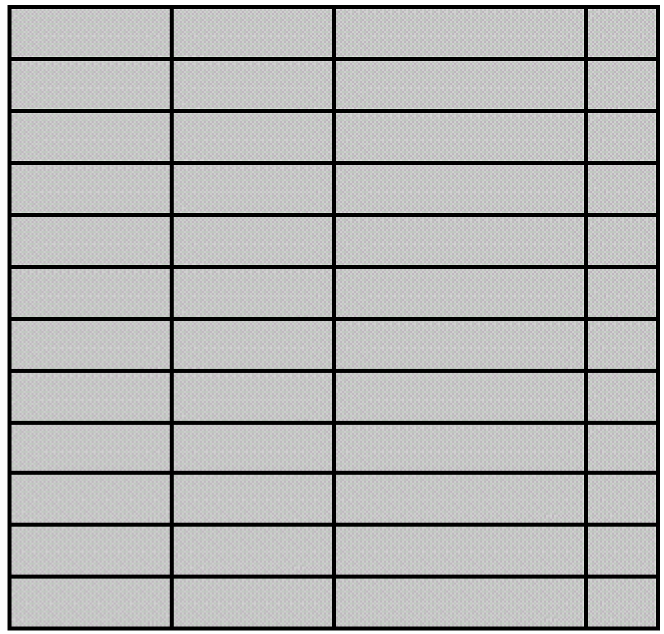
(a) Pile File

File Organization

- Sequential File
  - fixed format used for records
  - records are of the same length
  - same number of fixed-length fields in a particular order
  - one particular field is the key field
    - uniquely identifies the record
    - records are stored in key sequence
  - typically used in batch applications
    - involve the processing of all the records

File Organization

- Sequential File
  - poor performance for interactive applications
    - queries and/or updates of individual records
  - additions to the sequential file
    - new records are placed in a log file or transaction file
    - batch update is performed to merge the log file with the master file
  - an alternative is to organize the sequential file as a linked list



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

(b) Sequential File

File Organization

- Indexed Sequential File
  - an approach to overcome the disadvantages of the sequential file
    - records are organized in sequence based on a key field
    - an index to the file to support random access
    - overflow table
      - similar to the log file

File Organization

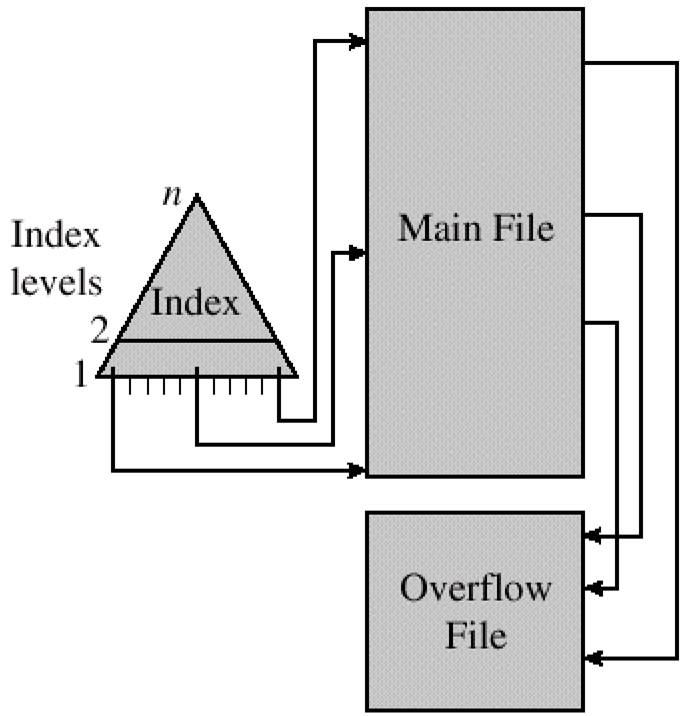
- Indexed Sequential File
  - index provides a lookup capability to quickly reach the vicinity of the desired record
    - contains key field and a pointer to the main file
    - index is searched to find highest key value that is equal or less than the desired key value
    - search continues in the 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

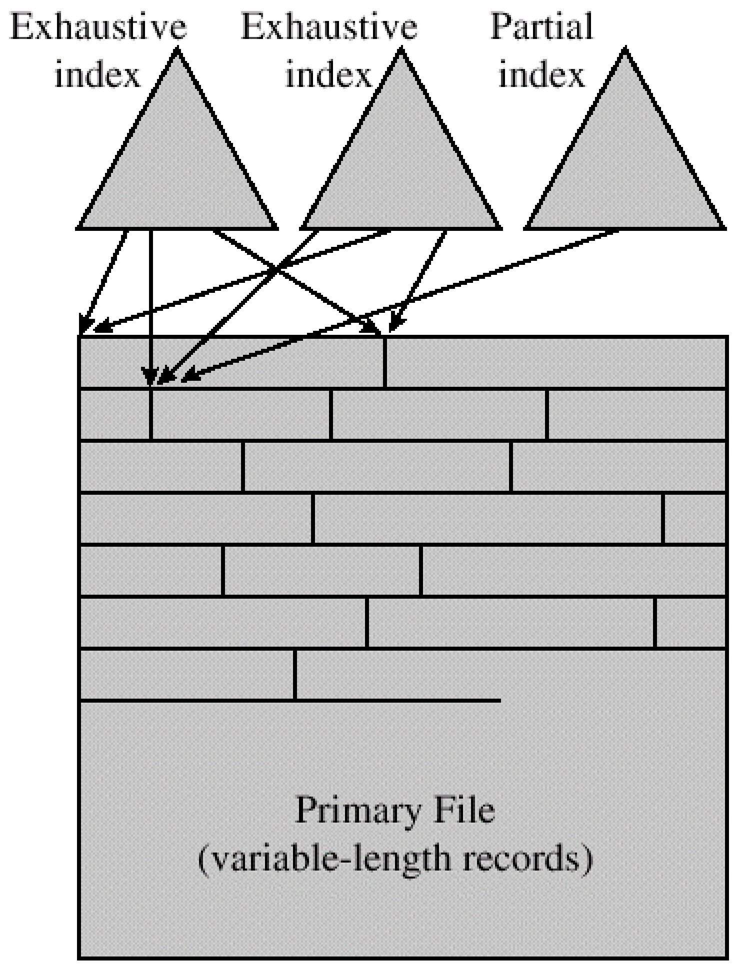
- Indexed Sequential File
  - new records are added to an overflow file
  - record in main file that precedes it is updated to contain a pointer to the new record
  - the overflow file is merged with the main file during a batch update



(c) Indexed Sequential File

File Organization

- Indexed File
  - limitation of indexed sequential file
    - effective processing is limited to that which is based on a single field of the file
  - uses multiple indexes for different key fields
  - when a new record is added to the main file, all of the index files must be updated
  - used in applications where timeliness of information is critical
    - airline reservation systems



(d) Indexed File

File Organization

- The Direct or Hashed File
  - key field required for each record
  - hashing on the key value to get the location of the record
  - used where
    - rapid access is required
    - fixed length records are used

File Organization

The Direct, or Hashed, File

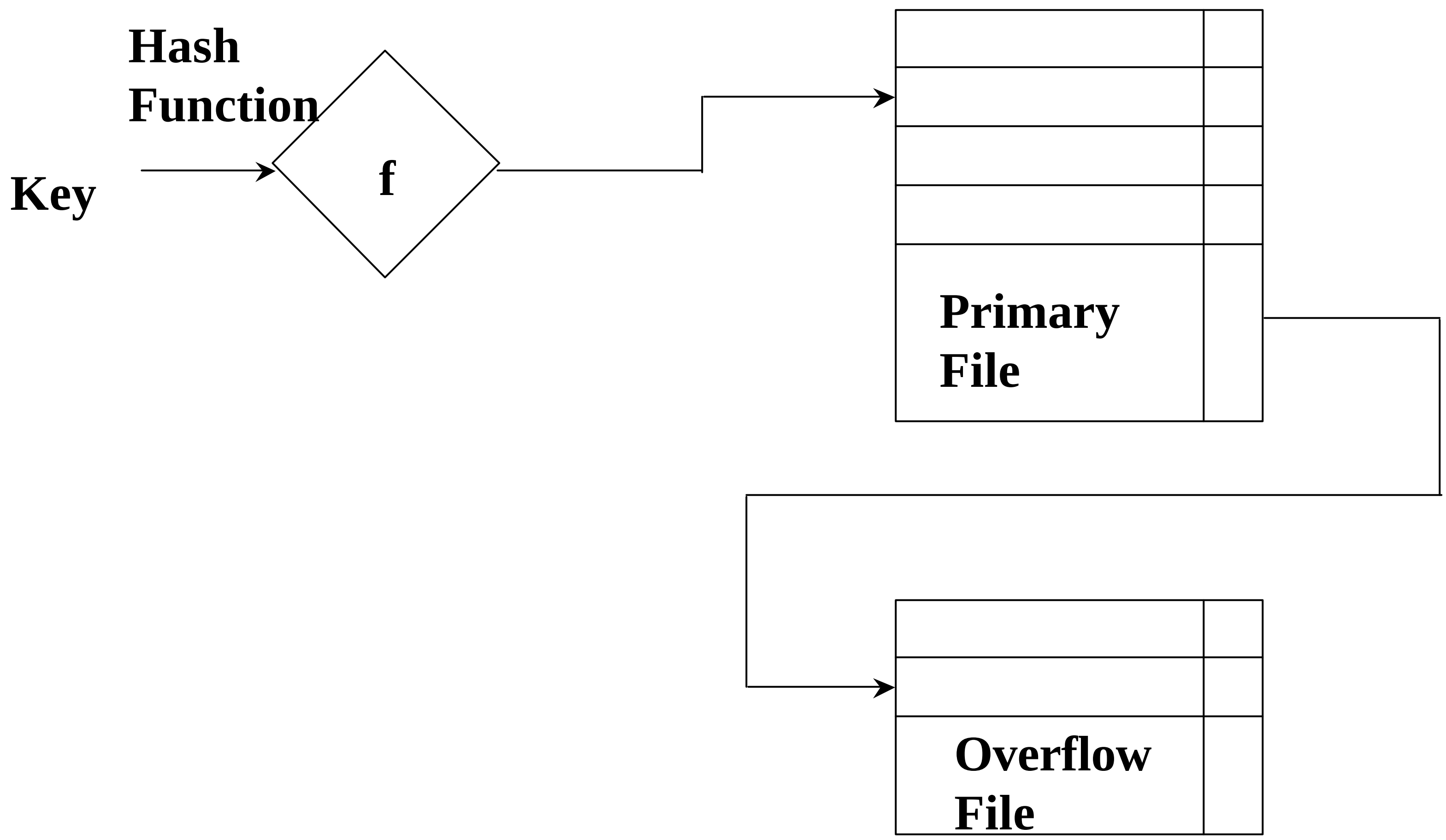


Table 12.1 Grades of performance for five basic file organizations

File Method	Space: Attributes Variable	Space: Attributes Fixed	Update: Record Size Equal	Update: Record Size Greater	Retrieval: Single record	Retrieval: Subset	Retrieval: 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 owned by the operating system
- Provides mapping between file names and the files themselves

Simple Structure for a Directory

- 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
  - the problem is much worse in shared system

Two-level Scheme for a Directory

- 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

Hierarchical, or Tree-Structured Directory

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries
- Files can be located by following a path from the root, or master, directory down various branches
  - this is the pathname for the file

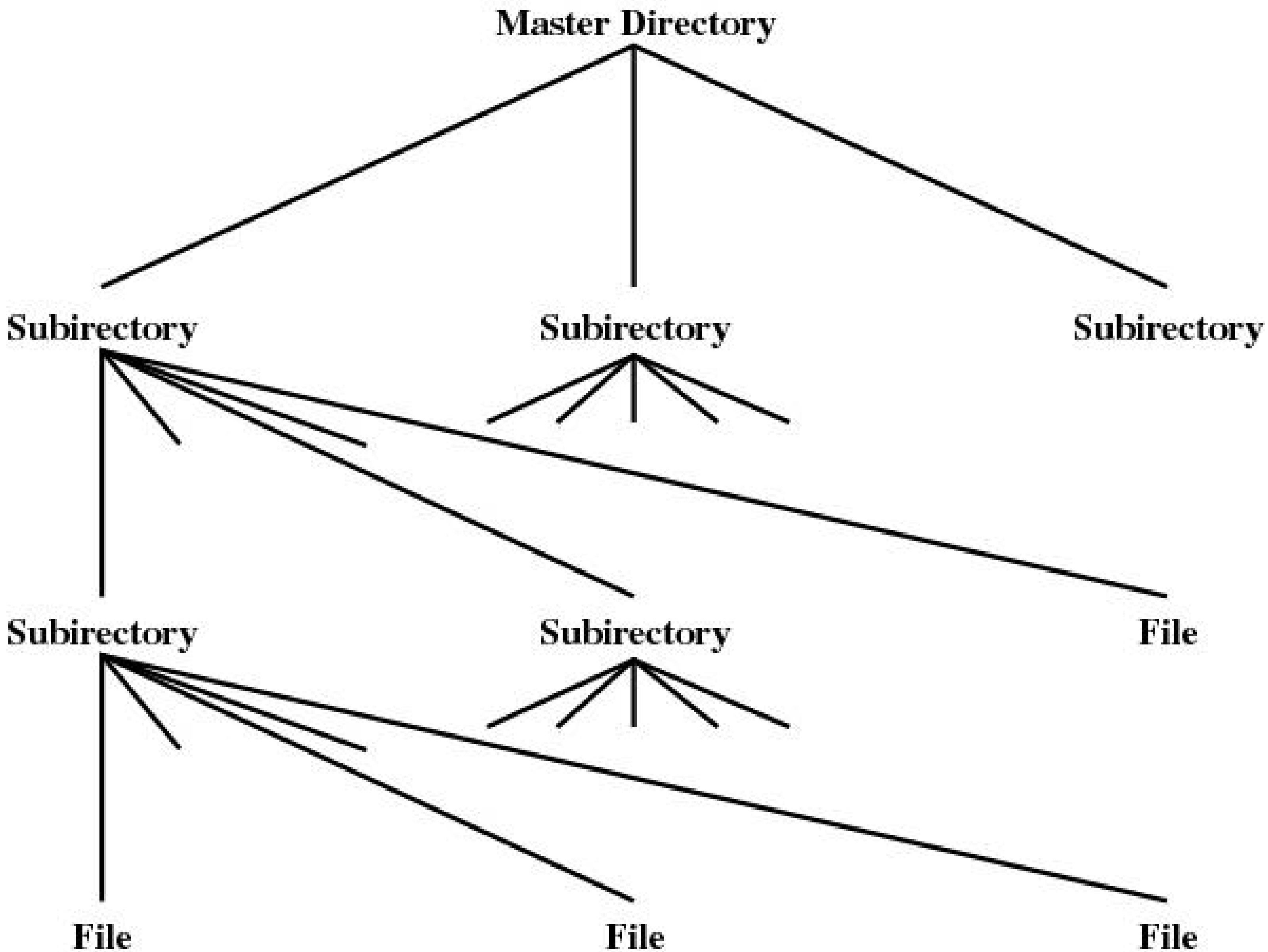


Figure 12.4 Tree-Structured Directory

Hierarchical, or Tree-Structured Directory

- Can have several files with the same file name as long as they have unique path names
- Current directory is the working directory
- Files can be referenced relative to the working directory

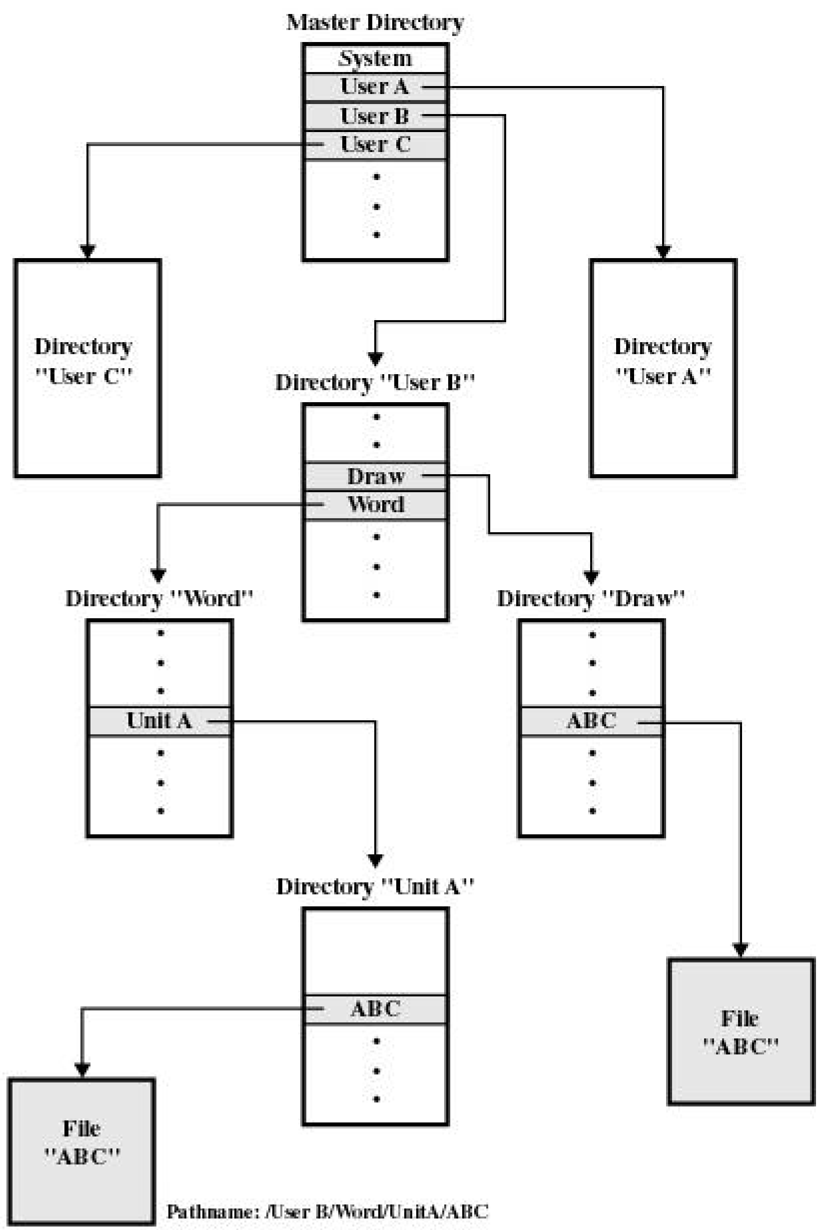


Figure 12.5 Example of Tree-Structured Directory

File Sharing

- Typical multiuser system allows 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 a 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
  - useful in collecting data from different users

Access Rights

- Updating
  - the user can modify, delete, and add to the file's data
- Changing protection
  - user can change the 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

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 simultaneous access

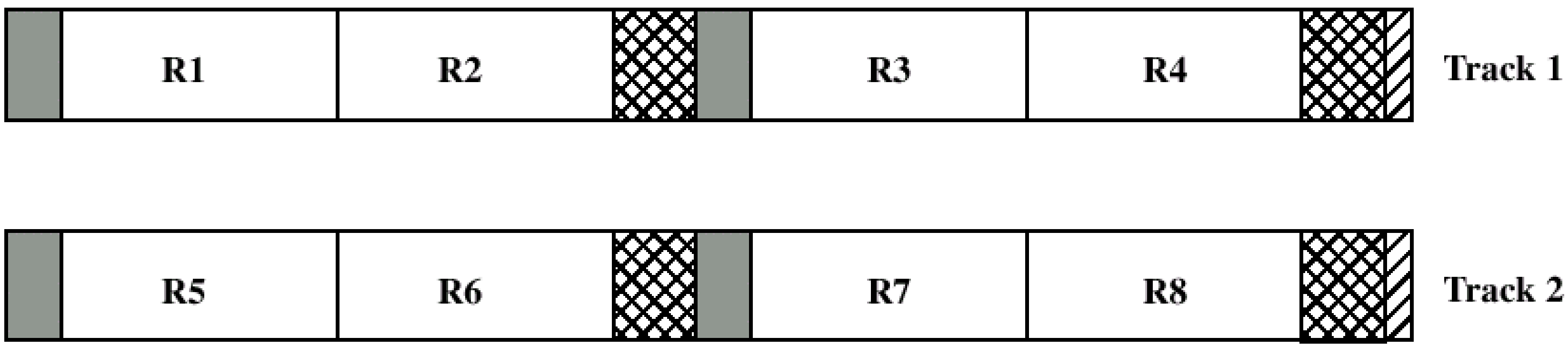
Record Blocking

- Records and Blocks
  - records are the logical unit of access of a file
  - blocks are the unit of I/O with secondary storage
- Issues to consider
  - fixed or variable block
  - size of a block
    - if a file is processed sequentially, larger blocks can reduce number of I/O operations
    - if records are accessed randomly, larger blocks result in the unnecessary transfer of unused records

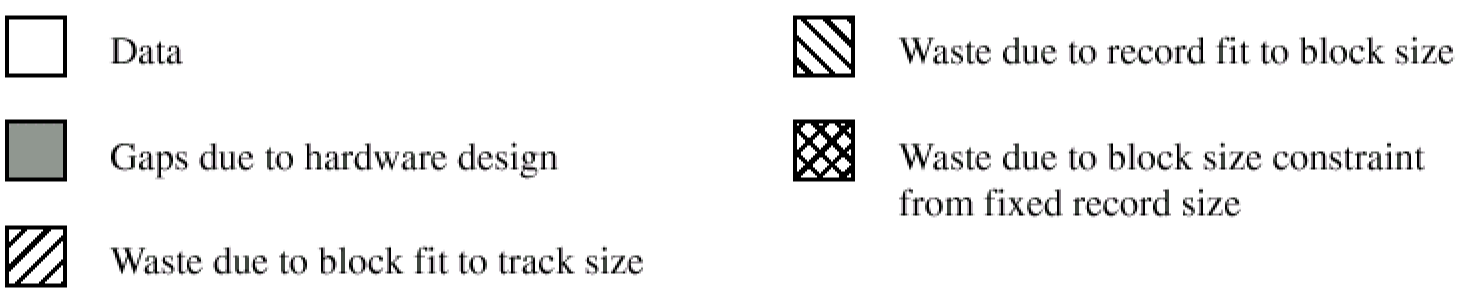
Fixed Blocking

- Fixed-length records are used
- Integral number of records are stored in a block
- There may be unused space at the end of each block
  - internal fragmentation
- Commonly used for sequential files

Fixed Blocking



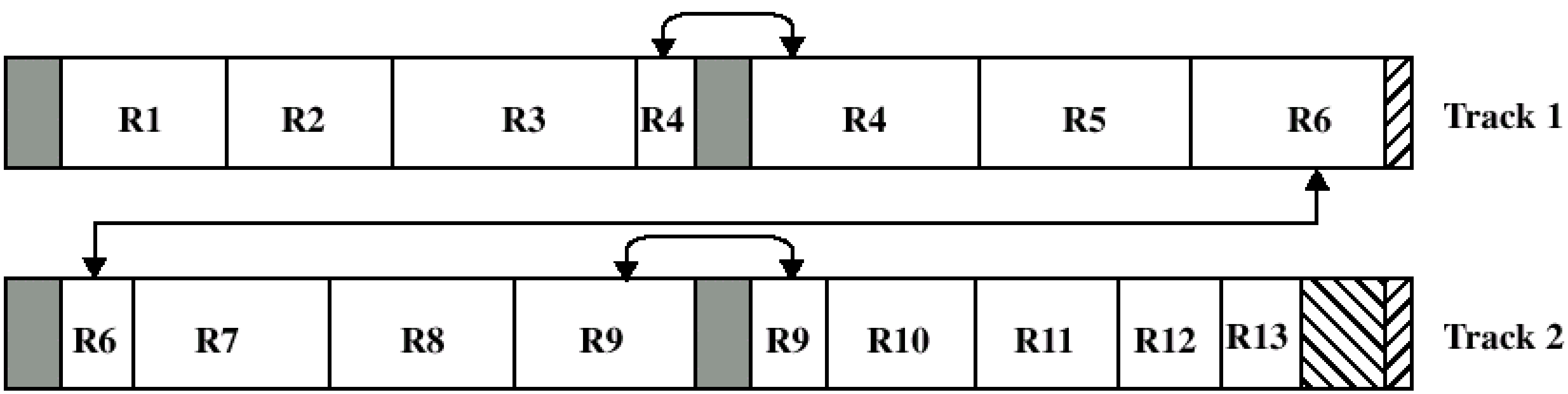
Fixed Blocking



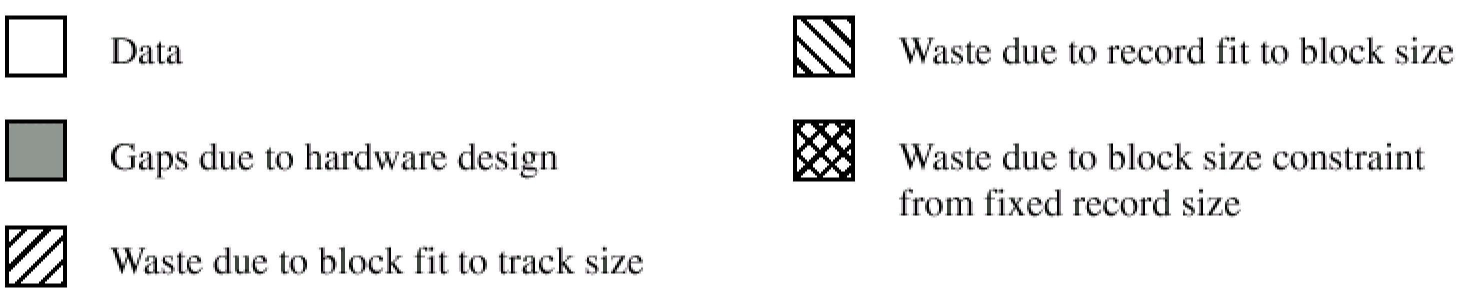
Variable Blocking : Spanned

- Variable-length records are used
- Records are packed into blocks with no unused space
  - some records may span two blocks
  - it is indicated by a pointer to the successor block
- Efficient use of storage and no limit on the size of records
- But difficult to implement

Variable Blocking : Spanned



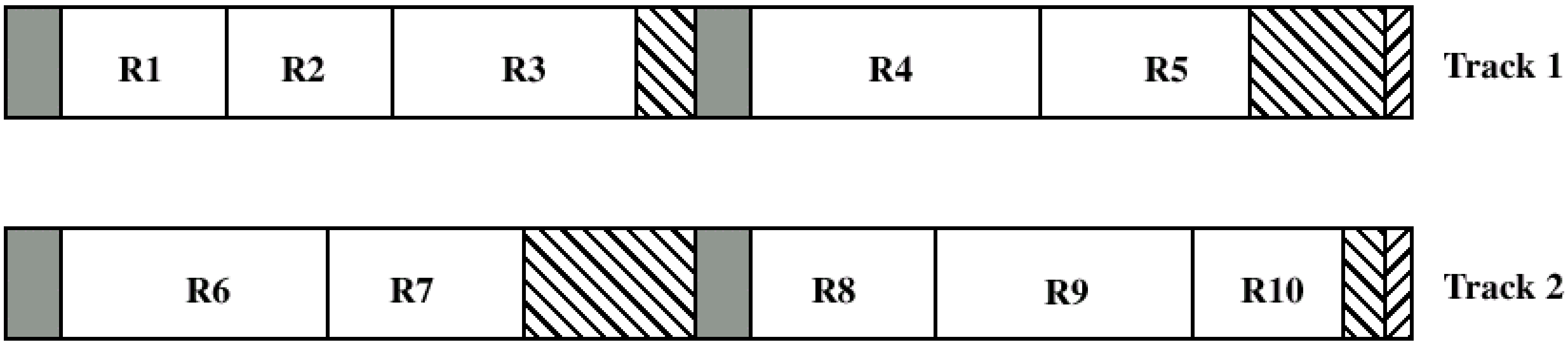
Variable Blocking: Spanned



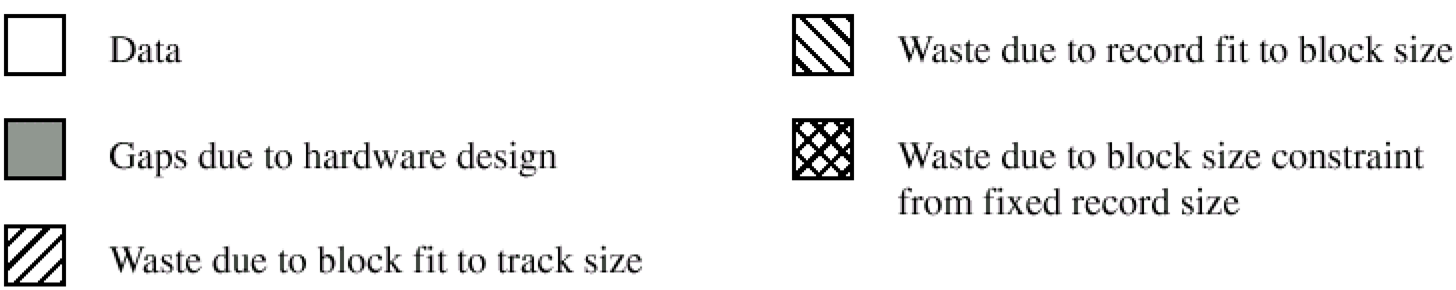
Variable Blocking : Unspanned

- Variable-length records are used
- Spanning is not employed
  - there is a wasted space in most blocks
- Results in wasted space and limits record size

Variable Blocking : Unspanned



Variable Blocking: Unspanned



Secondary Storage Management

- A file consists of a collection of blocks
- Management issues
  - File allocation
    - space on secondary storage must be allocated to files
  - Free space management
    - must keep track of the space available for allocation

File Allocation

- Issues to consider
  - preallocation VS dynamic allocation
  - unit of allocation
  - file allocation table(FAT)
    - a data structure that is used to keep track of the space assigned to a file

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
- So there are advantages to the use of dynamic allocation

Portion Size

- Tradeoff between user's view efficiency vs overall system efficiency
  - Contiguity of space increases performance
  - Large number of small portions increases the size of management tables
  - Fixed-size simplifies the reallocation of space
  - Variable-size minimizes waste of unused storage

Portion Size

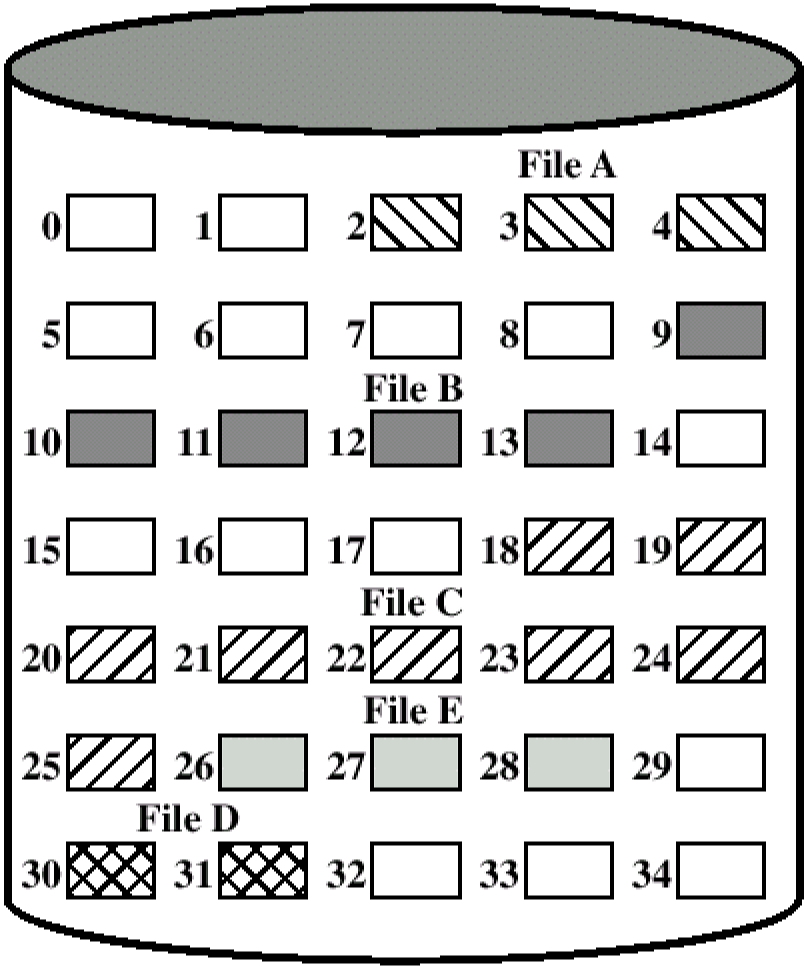
- Two major alternatives
  - Variable, large contiguous portions
    - avoids wasted space
    - file allocation tables are small
    - space is hard to reuse
  - Blocks
    - provides greater flexibility
    - requires large allocation tables
    - contiguity is abandoned

Methods of File Allocation

- File allocation methods
  - Contiguous allocation
  - Chained allocation
  - Indexed allocation

Methods of File Allocation

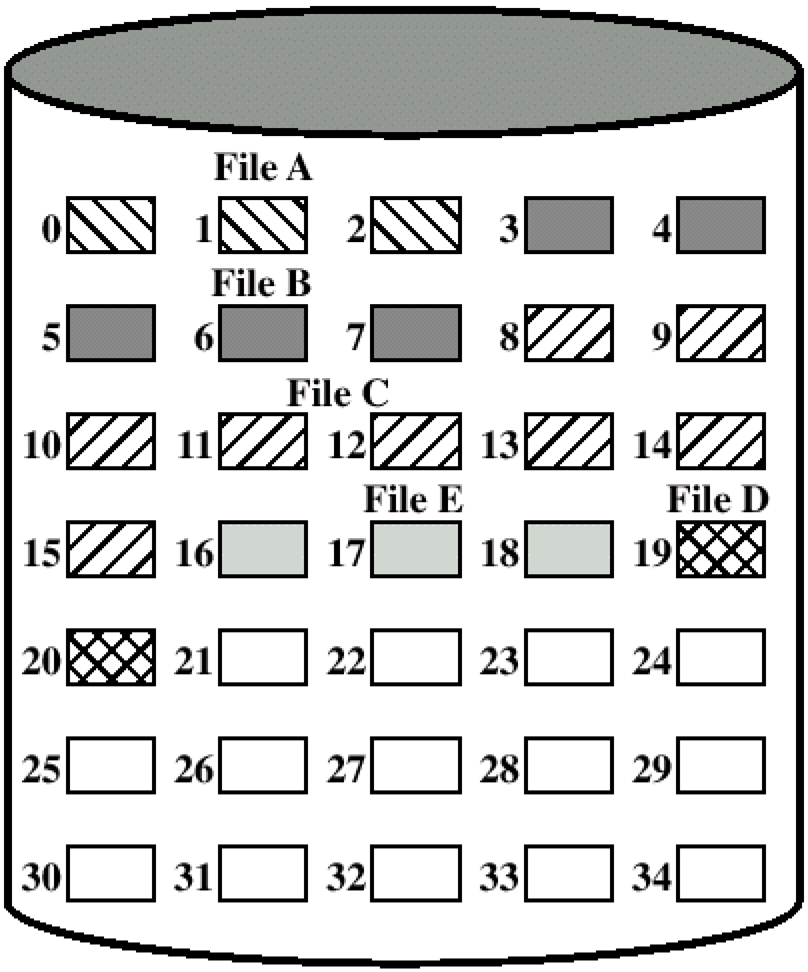
- Contiguous allocation
  - a single contiguous set of blocks is allocated to a file at the time of creation
    - preallocation using variable-size portions
  - only a single entry in the file allocation table
    - starting block and length of the file
  - external fragmentation will occur
    - difficult to find contiguous blocks of sufficient length
    - compaction is needed from time to time



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)

Methods of File Allocation

- Chained allocation
  - allocation on an individual block basis
  - 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
  - any free block can be added to the chain
  - no accommodation of the principle of locality
    - some systems periodically consolidate files

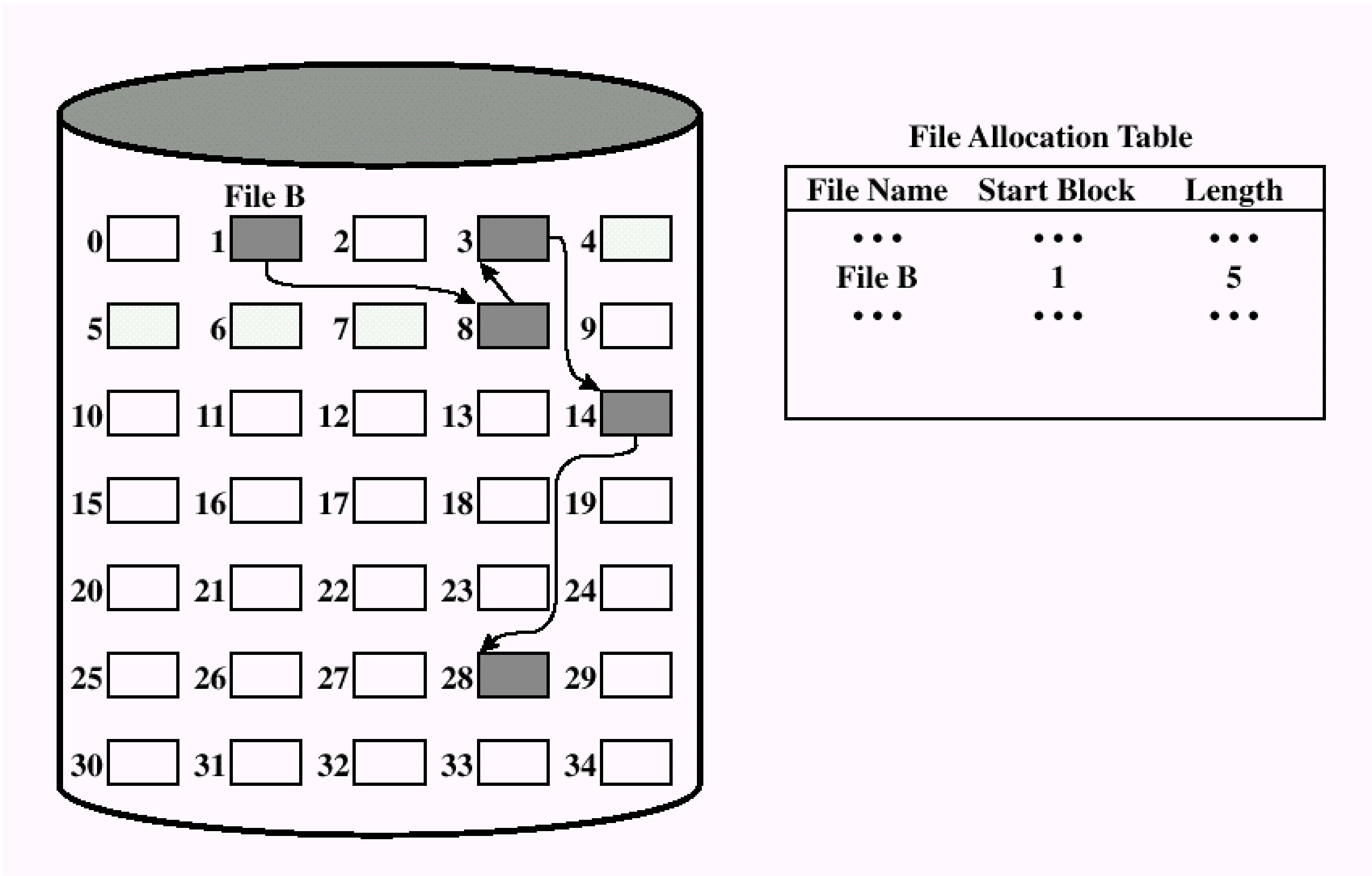
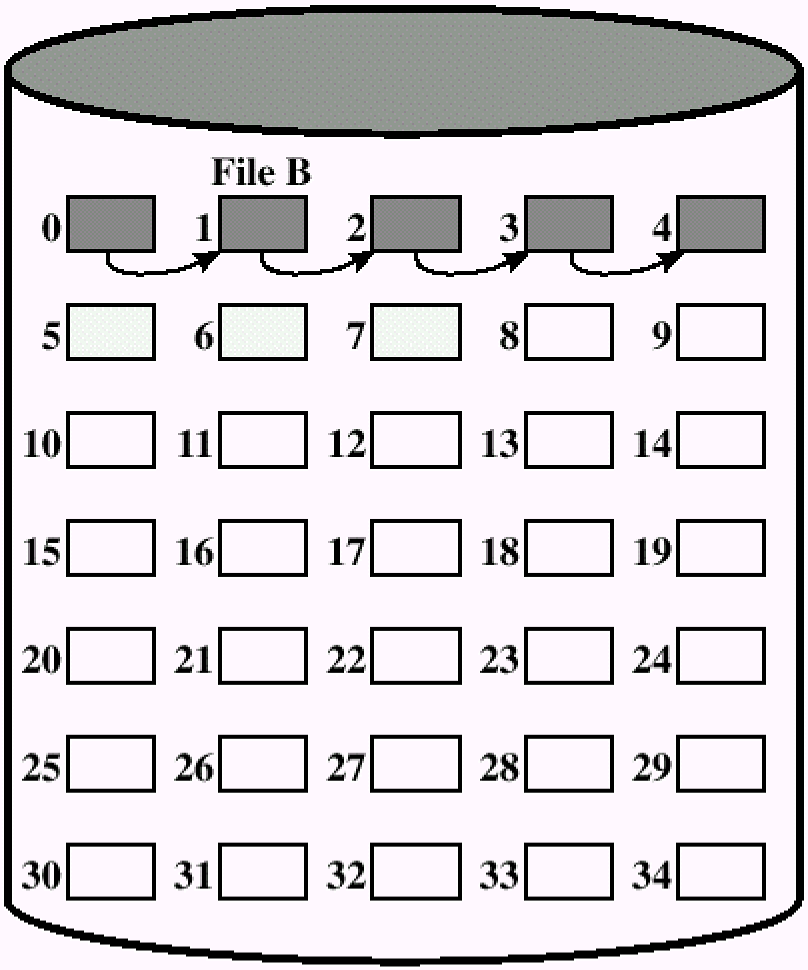


Figure 12.12 Chained allocation



File Allocation Table

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

Figure 12.10 Chained allocation (after consolidation)

Methods of File Allocation

- Indexed allocation : Unix file system
  - the file allocation table contains block number for the index
    - the index block has one entry for each portion allocated to the file
  - allocation may be either fixed-size or variable-size

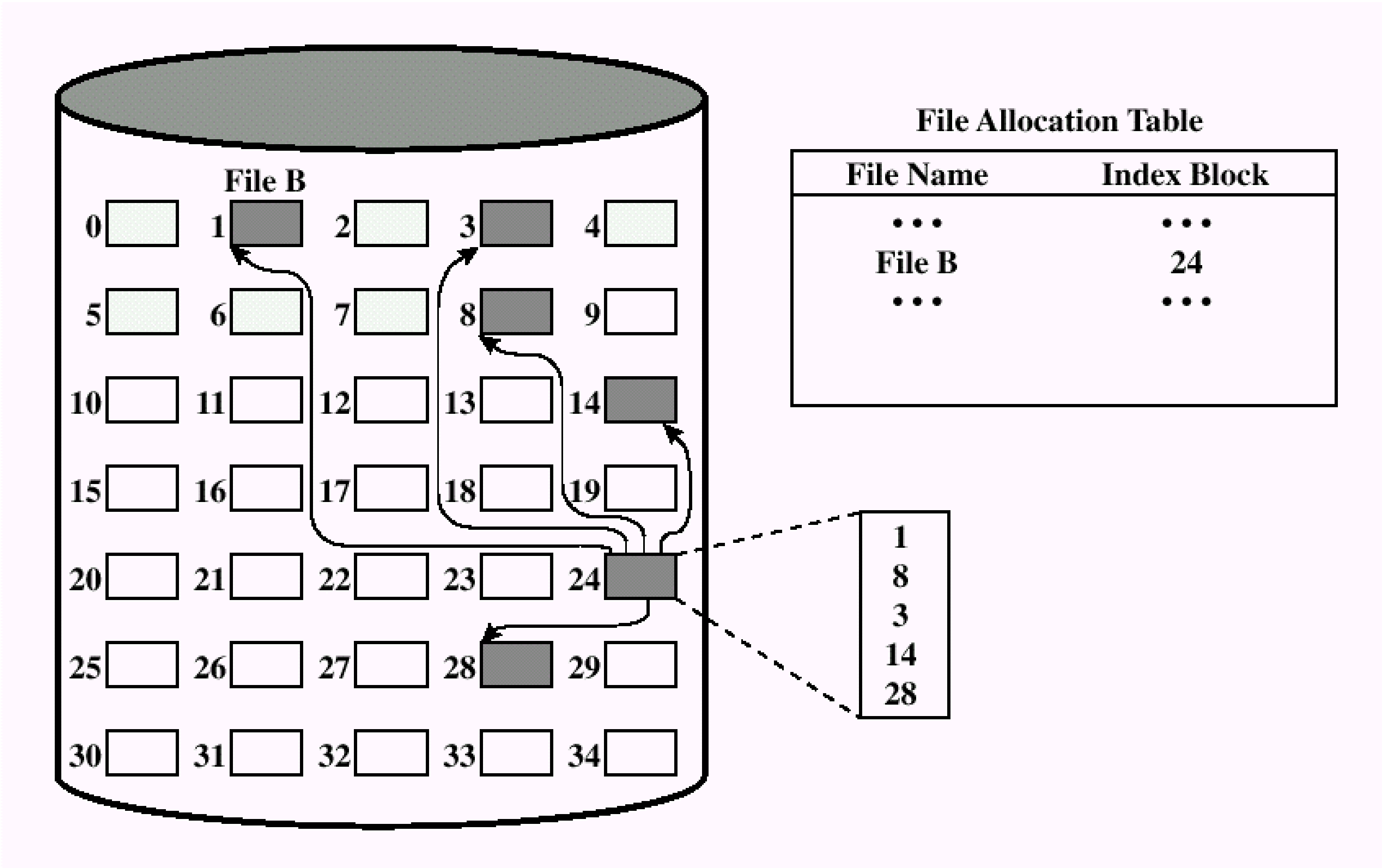


Figure 12.11 Indexed allocation with block portions

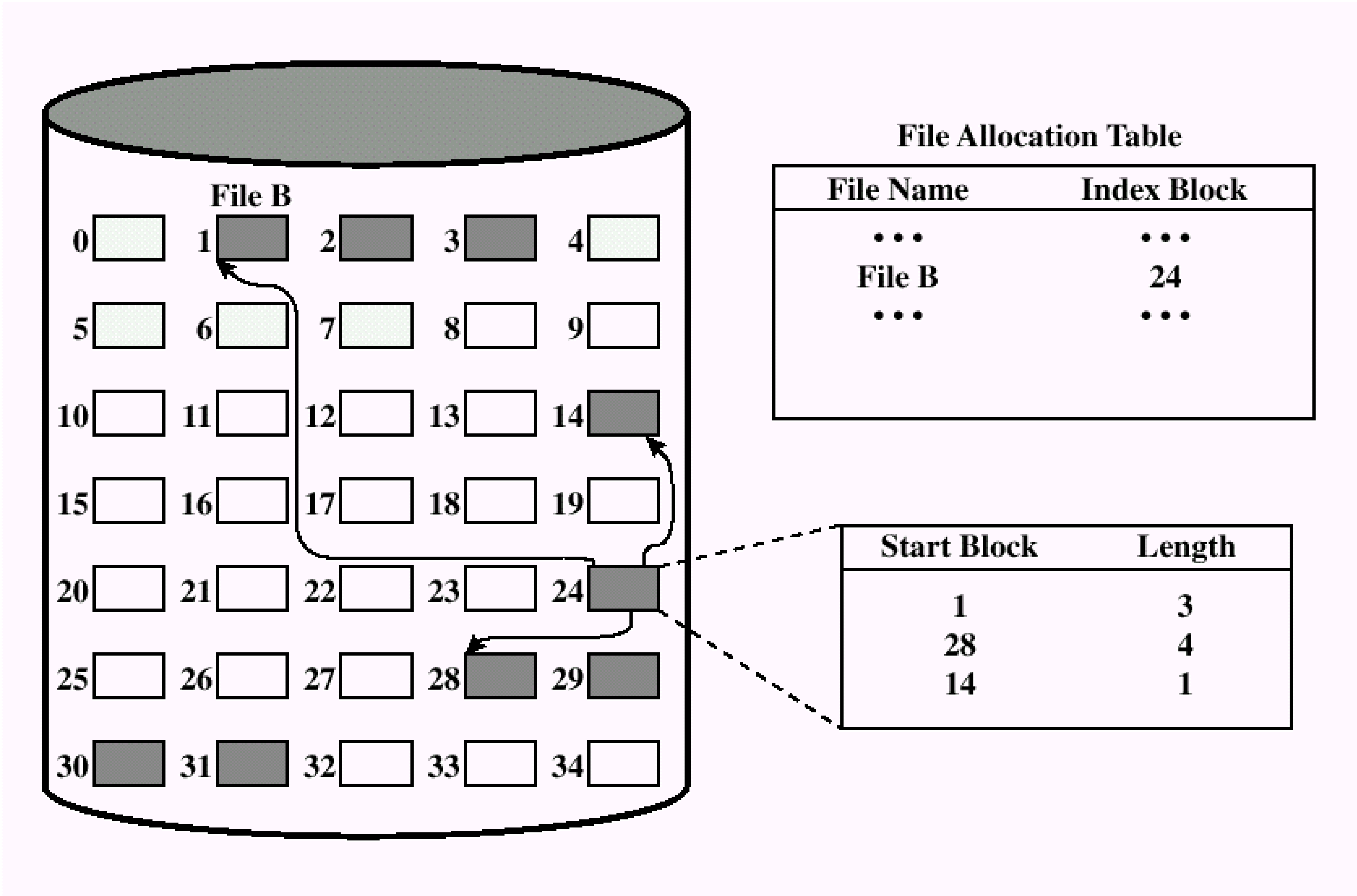


Figure 12.12 Indexed allocation with variable-length portions

Table 12.3 File Allocation Methods

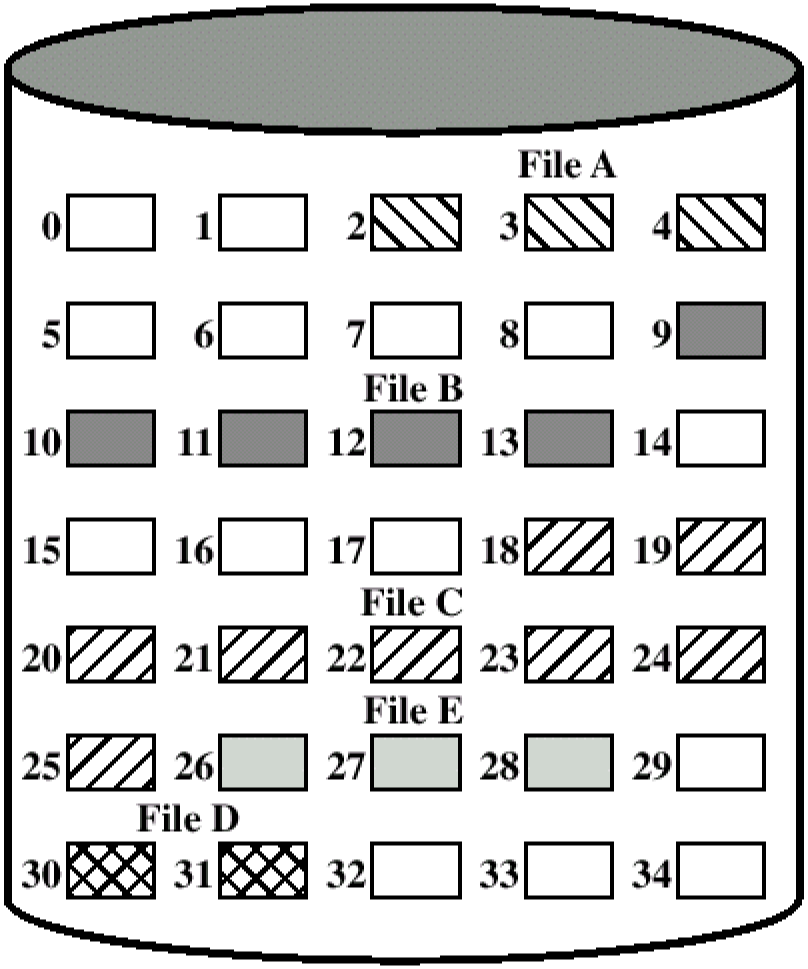
	Contiguous	Chained	Indexed	
Pre-Allocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table size	One entry	One entry	Large	Medium

Free Space Management

- The space that is not currently allocated to any file must be managed
- Disk allocation table
  - manages what blocks on the disk are free
- Methods for free space management
  - Bit tables
  - Chained free portions
  - Indexing
  - Free block list

Bit Tables

- A vector containing one bit for each block on the disk is used
  - entry of 0 corresponds to a free block
  - an example
    - 001110000111110000111111111111011000
- easy to find free blocks
- it is as small as possible



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

Chained Free Portions

- Free portions are chained by using a pointer
  - no need for a disk allocation table
- Every time a block is allocated, pointer needs to be adjusted
  - if many individual blocks need to be allocated at one time, this greatly slows down the process

Indexing

- Index table is used
  - one entry in the table for every free portion on the disk
- Provides efficient support for all of the file allocation methods

Free Block List

- Each block is assigned a number
- Numbers of all free blocks are maintained
  - assuming 32 bits for a block number, size of the free block list is 32 times the size of the bit table
- Only a small part of the list may reside in main memory
  - stack or FIFO queue can be used for this purpose

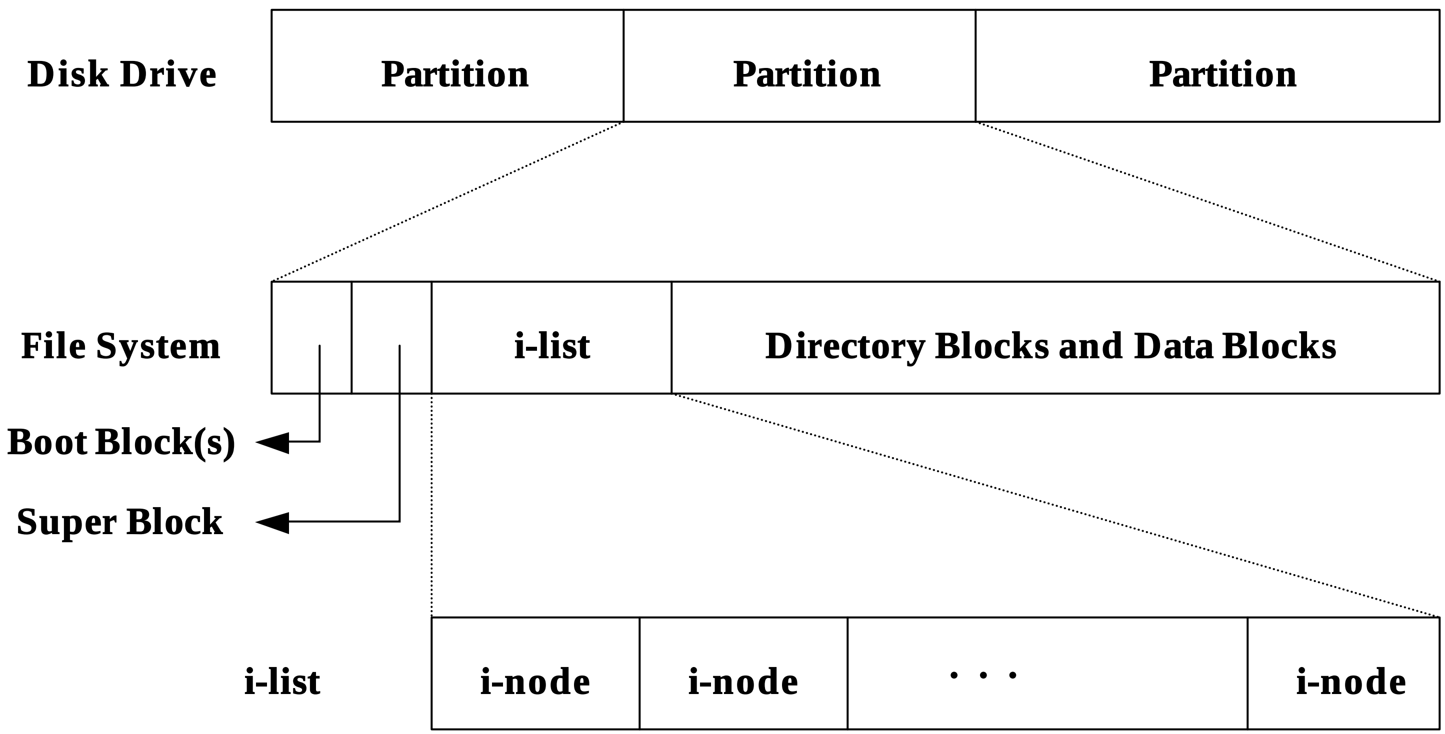
Reliability

- Consistency problem of disk allocation and file allocation table between main memory and disk
  - due to the fact that the system maintained a copy of the disk allocation table and file allocation table in main memory for efficiency

UNIX File Management

- Files are streams of bytes
- Types of files
  - ordinary - contents entered by user or program
  - directory - contains list of file names and pointers to inodes (index nodes)
  - special - used to access peripheral devices
  - named - named pipes

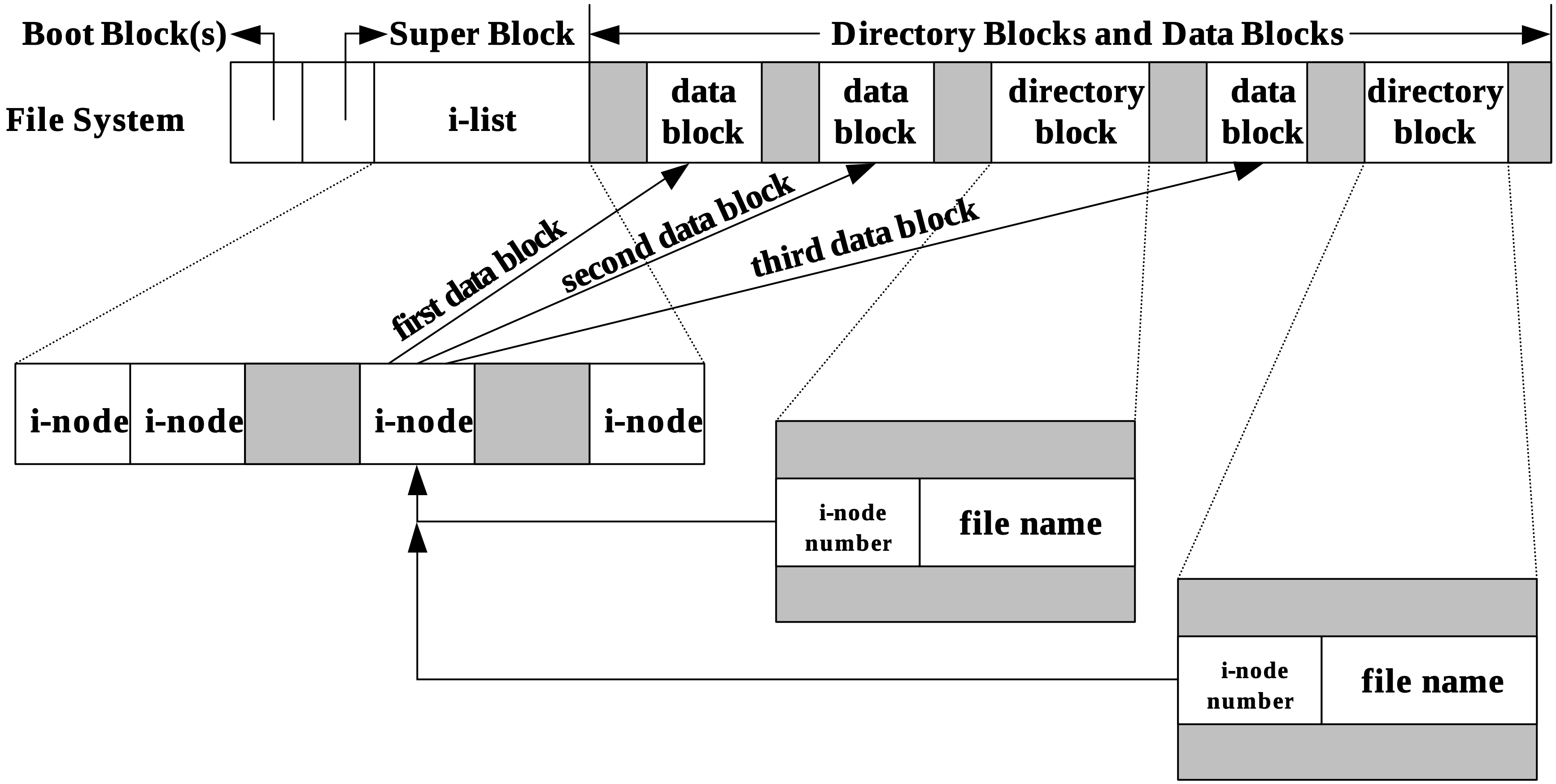
UNIX File System



UNIX File System: i-node

- File owner, group owner identifier
- File type
- File access permission
- Access, modified time
- Number of links to the file
- File size
- Table of contents for the disk addresses of data

UNIX File System in More Detail



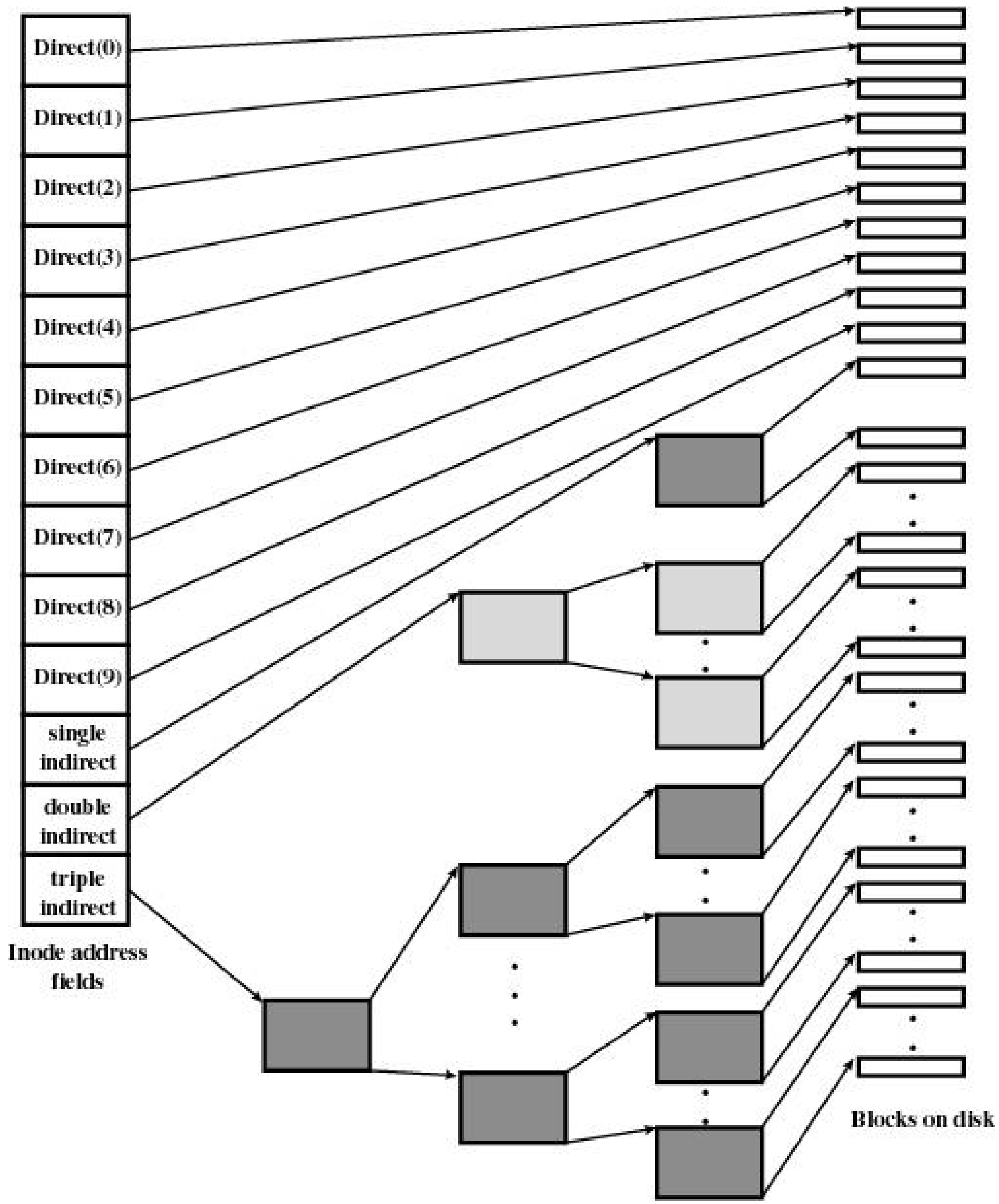


Figure 12.13 UNIX Block Addressing Scheme

Table 12.5 Capacity of a UNIX File

Level	Number of Blocks	Number of Bytes
Direct	10	10K
Single Indirect	256	256K
Double Indirect	$256 \times 256 = 65K$	65M
Triple Indirect	$256 \times 65K = 16M$	16G

Windows 2000 File System

- Key features of NTFS
  - Recoverability
  - Security
  - Large disks and large files
  - Multiple data streams
  - General indexing facility

Windows NT File System

- Sector - smallest unit of storage on a disk
- Cluster - one or more contiguous sectors
- Volume - logical partition on a disk

Table 12.6 Windows NTFS Partition and Cluster Sizes

Volume Size	Sectors per Cluster	Cluster Size
≤ 512 Mbyte	1	512 bytes
512 Mbyte - 1 Gbyte	2	1K
1 Gbyte - 2 Gbyte	4	2K
2 Gbyte - 4 Gbyte	8	4K
4 Gbyte - 8 Gbyte	16	8K
8 Gbyte - 16 Gbyte	32	16K
16 Gbyte - 32 Gbyte	64	32K
> 32 Gbyte	128	64K

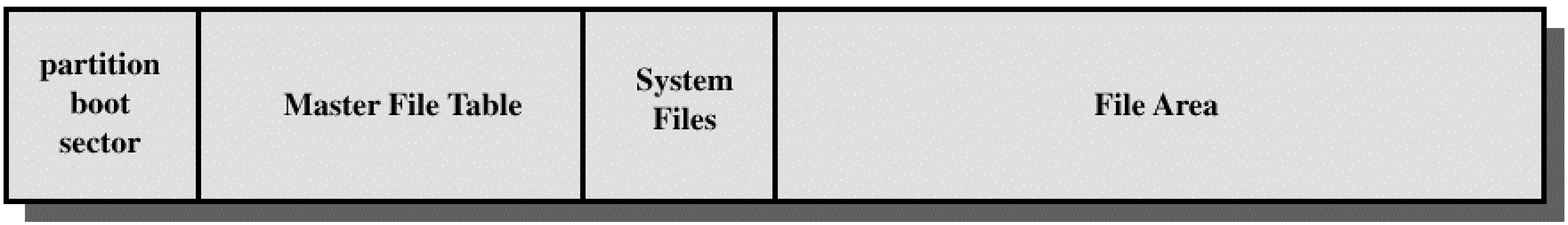


Figure 12.14 NTFS Volume Layout