

Chapter 13 – File and Database Systems

Outline

- 13.1 Introduction
- 13.2 Data Hierarchy
- 13.3 Files
- 13.4 File Systems
 - 13.4.1 Directories
- 13.4. Metadata
- 13.4. Mounting
- 13.5 File Organization
- 13.6 File Allocation
 - 13.6.1 Contiguous File Allocation
 - 13.6.2 Linked-List Noncontiguous File Allocation
 - 13.6.3 Tabular Noncontiguous File Allocation
 - 13.6.4 Indexed Noncontiguous File Allocation
- 13.7 Free Space Management
- 13.8 File Access Control
 - 13.8.1 Access Control Matrix
 - 13.8.2 Access Control by User Classes
- 13.9 Data Access Techniques

© 2004 Deitel & Associates, Inc. All rights reserved.



Chapter 13 – File and Database Systems

Outline (continued)

- 13.10 Data Integrity Protection
 - 13.10.1 Backup and Recovery
 - 13.10.2 Data Integrity and Log-Structured File Systems
- 13.11 File Servers and Distributed Systems
- 13.12 Database Systems
 - 13.12.1 Advantages of Database Systems
 - 13.12.2 Data Access
 - 13.12.3 Relational Database Model
 - 13.12.4 Operating Systems and Database Systems

© 2004 Deitel & Associates, Inc. All rights reserved.



Objectives

- After reading this chapter, you should understand:
 - the need for file systems.
 - files, directories and the operations that can be performed on them.
 - organizing and managing a storage device's data and free space.
 - controlling access to data in a file system.
 - backup, recovery and file system integrity mechanisms.
 - database systems and models.

© 2004 Deitel & Associates, Inc. All rights reserved.



13.1 Introduction

- Files
 - Named collection of data that is manipulated as a unit
 - Reside on secondary storage devices
- Operating systems can create an interface that facilitates navigation of a user's files
 - File systems can protect such data from corruption or total loss from disasters
 - Systems that manage large amounts of shared data can benefit from databases as an alternative to files

© 2004 Deitel & Associates, Inc. All rights reserved.



13.2 Data Hierarchy

- Information is stored in computers according to a data hierarchy.
- Lowest level of data hierarchy is composed of bits
 - Bit patterns represent all data items of interest in computer systems

© 2004 Deitel & Associates, Inc. All rights reserved.



13.2 Data Hierarchy

- Next level in the data hierarchy is fixed-length patterns of bits such as bytes, characters and words
 - Byte: typically 8 bits
 - Word: the number of bits a processor can operate on at once
 - Characters map bytes (or groups of bytes) to symbols such as letters, numbers, punctuation and new lines
 - Three most popular character sets in use today: ASCII, EBCDIC and Unicode
 - Field: a group of characters
 - Record: a group of fields
 - File: a group of related records

© 2004 Deitel & Associates, Inc. All rights reserved.



13.2 Data Hierarchy

- Highest level of the data hierarchy is a file system or database
- A volume is a unit of data storage that may hold multiple files

© 2004 Deitel & Associates, Inc. All rights reserved.



13.3 Files

- File: a named collection of data that may be manipulated as a unit by operations such as:
 - Open
 - Close
 - Create
 - Destroy
 - Copy
 - Rename
 - List

© 2004 Deitel & Associates, Inc. All rights reserved.



13.3 Files

- Individual data items within a file may be manipulated by operations like:
 - Read
 - Write
 - Update
 - Insert
 - Delete
- File characteristics include:
 - Location
 - Accessibility
 - Type
 - Volatility
 - Activity
- Files can consist of one or more records

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4 File Systems

- File systems
 - Organize files and manages access to data
 - Responsible for file management, auxiliary storage management, file integrity mechanisms and access methods
 - Primarily are concerned with managing secondary storage space, particularly disk storage

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4 File Systems

- File system characteristics
 - Should exhibit device independence:
 - Users should be able to refer to their files by symbolic names rather than having to use physical device names
 - Should also provide backup and recovery capabilities to prevent either accidental loss or malicious destruction of information
 - May also provide encryption and decryption capabilities to make information useful only to its intended audience

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

- Directories:
 - Files containing the names and locations of other files in the file system, to organize and quickly locate files
- Directory entry stores information such as:
 - File name
 - Location
 - Size
 - Type
 - Accessed
 - Modified and creation times

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

Figure 13.1 Directory file contents example.

<i>Directory Field</i>	<i>Description</i>
Name	Character string representing the file's name.
Location	Physical block or logical location of the file in the file system (i.e., a pathname).
Size	Number of bytes consumed by the file.
Type	Description of the file's purpose (e.g., data file or directory file).
Access time	Time the file was last accessed.
Modified time	Time the file was last modified.
Creation time	Time the file was created.



13.4.1 Directories

- Single-level (or flat) file system:
 - Simplest file system organization
 - Stores all of its files using one directory
 - No two files can have the same name
 - File system must perform a linear search of the directory contents to locate each file, which can lead to poor performance



13.4.1 Directories

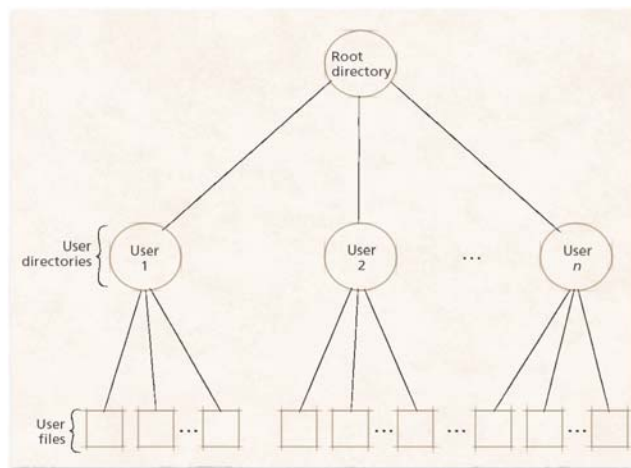
- Hierarchical file system:
 - A root indicates where on the storage device the root directory begins
 - The root directory points to the various directories, each of which contains an entry for each of its files
 - File names need be unique only within a given user directory
 - The name of a file is usually formed as the pathname from the root directory to the file

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

Figure 13.2 Two-level hierarchical file system.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

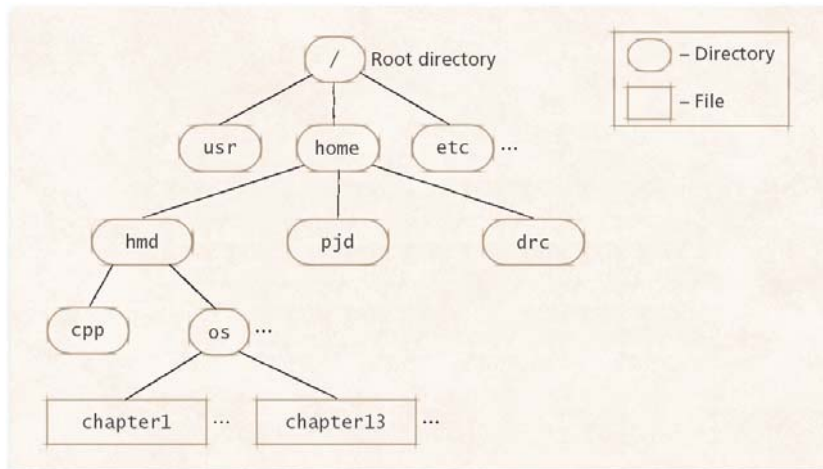
- Working directory
 - Simplifies navigation using pathnames
 - Enables users to specify a pathname that does not begin at the root directory (i.e., a relative path)
 - Absolute path (i.e., the path beginning at the root) = working directory + relative path

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

Figure 13.3 Example hierarchical file system contents.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

- Link: a directory entry that references a data file or directory located in a different directory
 - Facilitates data sharing and can make it easier for users to access files located throughout a file system's directory structure
 - Soft link: directory entry containing the pathname for another file
 - Hard link: directory entry that specifies the location of the file (typically a block number) on the storage device

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

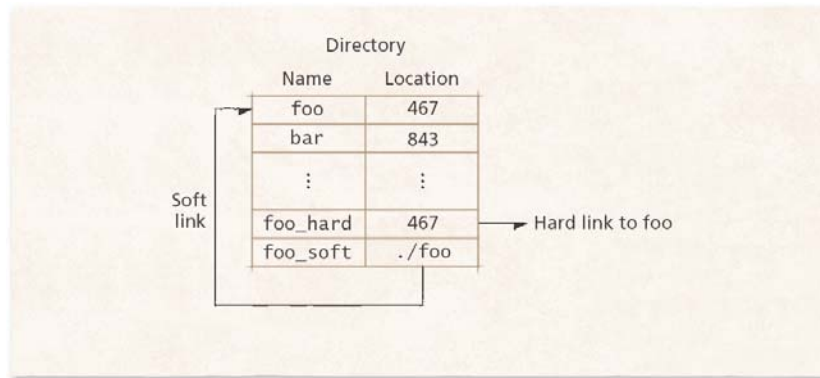
- Links (Cont.)
 - Because a hard link specifies a physical location of a file, it references invalid data when the physical location of its corresponding file changes
 - Because soft links store the logical location of the file in the file system, they do not require updating when file data is moved
 - However, if a user moves a file to different directory or renames the file, any soft links to that file are no longer valid

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.1 Directories

Figure 13.4 Links in a file system.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.2 Metadata

- Metadata
 - Information that protects the integrity of the file system
 - Cannot be modified directly by users
- Many file systems create a superblock to store critical information that protects the integrity of the file system
 - A superblock might contain:
 - The file system identifier
 - The location of the storage device's free blocks
 - To reduce the risk of data loss, most file systems distribute redundant copies of the superblock throughout the storage device

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.2 Metadata

- File open operation returns a file descriptor
 - A non-negative integer index into the open-file table
- From this point on, access to the file is directed through the file descriptor
- To enable fast access to file-specific information such as permissions, the open-file table often contains file control blocks, also called file attributes:
 - Highly system-dependent structures that might include the file's symbolic name, location in secondary storage, access control data and so on

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.3 Mounting

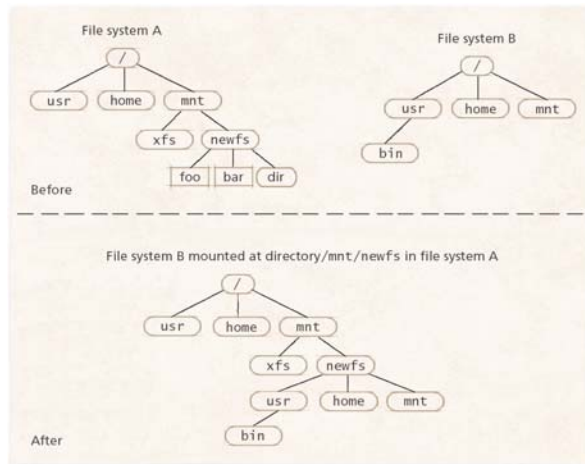
- Mount operation
 - Combines multiple file systems into one namespace so that they can be referenced from a single root directory
 - Assigns a directory, called the mount point, in the native file system to the root of the mounted file system
- File systems manage mounted directories with mount tables:
 - Contain information about the location of mount points and the devices to which they point
- When the native file system encounters a mount point, it uses the mount table to determine the device and type of the mounted file system
- Users can create soft links to files in mounted file systems but cannot create hard links between file systems

© 2004 Deitel & Associates, Inc. All rights reserved.



13.4.3 Mounting

Figure 13.5 Mounting a file system.



© 2004 Deitel & Associates, Inc. All rights reserved.

13.5 File Organization

- File organization: the manner in which the records of a file are arranged on secondary storage
- File organization schemes include:
 - Sequential
 - Direct
 - Indexed nonsequential
 - Partitioned

© 2004 Deitel & Associates, Inc. All rights reserved.

13.6 File Allocation

- File allocation
 - Problem of allocating and freeing space on secondary storage is somewhat like that experienced in primary storage allocation under variable-partition multiprogramming
 - Contiguous allocation systems have generally been replaced by more dynamic noncontiguous allocation systems
 - Files tend to grow or shrink over time
 - Users rarely know in advance how large their files will be

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.1 Contiguous File Allocation

- Contiguous allocation
 - Place file data at contiguous addresses on the storage device
 - Advantages
 - Successive logical records typically are physically adjacent to one another
 - Disadvantages
 - External fragmentation
 - Poor performance can result if files grow and shrink over time
 - If a file grows beyond the size originally specified and no contiguous free blocks are available, it must be transferred to a new area of adequate size, leading to additional I/O operations.

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.2 Linked-List Noncontiguous File Allocation

- Sector-based linked-list noncontiguous file allocation scheme:
 - A directory entry points to the first sector of a file
 - The data portion of a sector stores the contents of the file
 - The pointer portion points to the file's next sector
 - Sectors belonging to a common file form a linked list

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.2 Linked-List Noncontiguous File Allocation

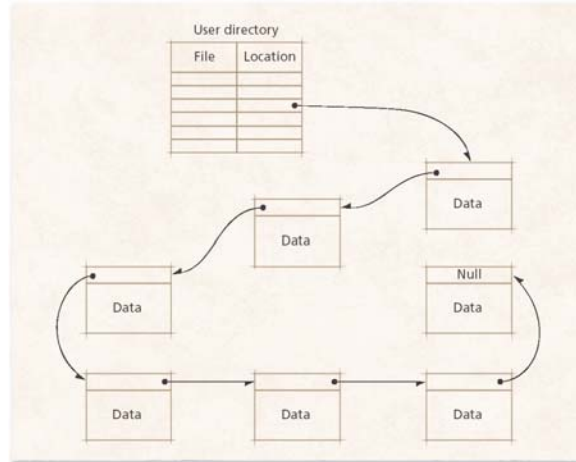
- When performing block allocation, the system allocates blocks of contiguous sectors (sometimes called extents)
- Block chaining
 - Entries in the user directory point to the first block of each file
 - File blocks contain:
 - A data block
 - A pointer to the next block

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.2 Linked-List Noncontiguous File Allocation

Figure 13.6 Noncontiguous file allocation using a linked list.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.2 Linked-List Noncontiguous File Allocation

- When locating a record
 - The chain must be searched from the beginning
 - If the blocks are dispersed throughout the storage device (which is normal), the search process can be slow as block-to-block seeks occur
- Insertion and deletion are done by modifying the pointer in the previous block

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.2 Linked-List Noncontiguous File Allocation

- Large block sizes
 - Can result in significant internal fragmentation
- Small block sizes
 - May cause file data to be spread across multiple blocks dispersed throughout the storage device
 - Poor performance as the storage device performs many seeks to access all the records of a file

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.3 Tabular Noncontiguous File Allocation

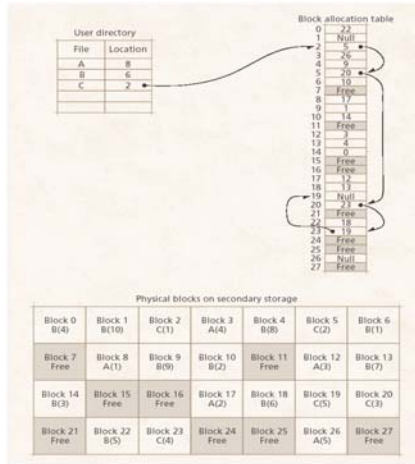
- Tabular noncontiguous file allocation
 - Uses tables storing pointers to file blocks
 - Reduces the number of lengthy seeks required to access a particular record
 - Directory entries indicate the first block of a file
 - Current block number is used as an index into the block allocation table to determine the location of the next block.
 - If the current block is the file's last block, then its block allocation table entry is null

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.3 Tabular Noncontiguous File Allocation

Figure 13.7 Tabular noncontiguous file allocation.



© 2004 Deitel & Associates, Inc. All rights reserved.

13.6.3 Tabular Noncontiguous File Allocation

- Pointers that locate file data are stored in a central location
 - The table can be cached so that the chain of blocks that compose a file can be traversed quickly
 - Improves access times
- To locate the last record of a file, however:
 - The file system might need to follow many pointers in the block allocation table
 - Could take significant time

© 2004 Deitel & Associates, Inc. All rights reserved.

13.6.3 Tabular Noncontiguous File Allocation

- When a storage device contains many blocks:
 - The block allocation table can become large and fragmented
 - Reduces file system performance
- A popular implementation of tabular noncontiguous file allocation is Microsoft's FAT file system

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.4 Indexed Noncontiguous File Allocation

- Indexed noncontiguous file allocation:
 - Each file has an index block or several index blocks
 - Index blocks contain a list of pointers that point to file data blocks
 - A file's directory entry points to its index block, which may reserve the last few entries to store pointers to more index blocks, a technique called chaining

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.4 Indexed Noncontiguous File Allocation

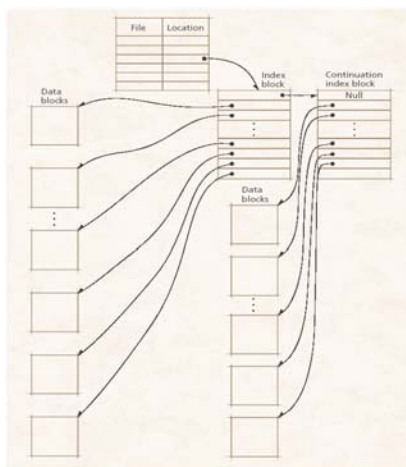
- Primary advantage of index block chaining over simple linked-list implementations:
 - Searching may take place in the index blocks themselves.
 - File systems typically place index blocks near the data blocks they reference, so the data blocks can be accessed quickly after their index block is loaded

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.4 Indexed Noncontiguous File Allocation

Figure 13.8 Index block chaining.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.4 Indexed Noncontiguous File Allocation

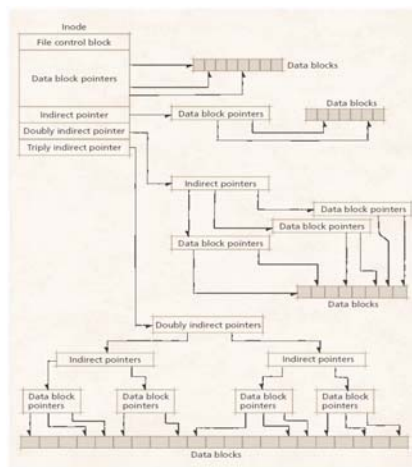
- Index blocks are called inodes (i.e., index nodes) in UNIX-based operating systems

© 2004 Deitel & Associates, Inc. All rights reserved.



13.6.4 Indexed Noncontiguous File Allocation

Figure 13.9 Inode structure.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.7 Free Space Management

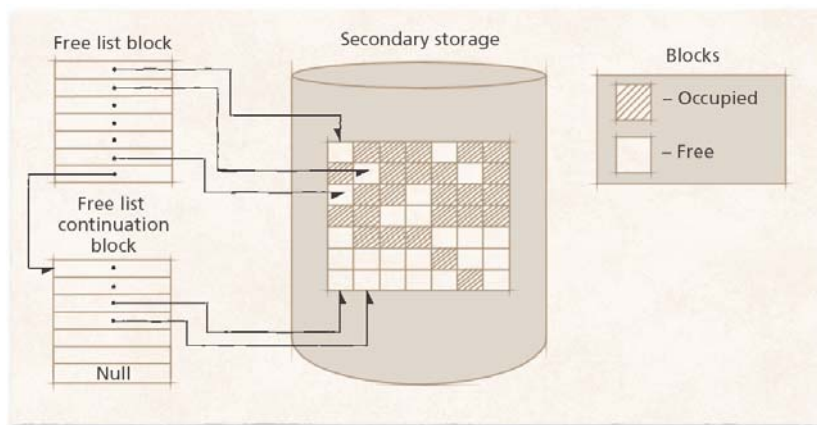
- Some systems use a free list to manage the storage device's free space
 - Free list: Linked list of blocks containing the locations of free blocks
 - Blocks are allocated from the beginning of the free list
 - Newly freed blocks are appended to the end of the list
- Low overhead to perform free list maintenance operations
- Files are likely to be allocated in noncontiguous blocks
 - Increases file access time

© 2004 Deitel & Associates, Inc. All rights reserved.



13.7 Free Space Management

Figure 13.10 Free space management using a free list.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.7 Free Space Management

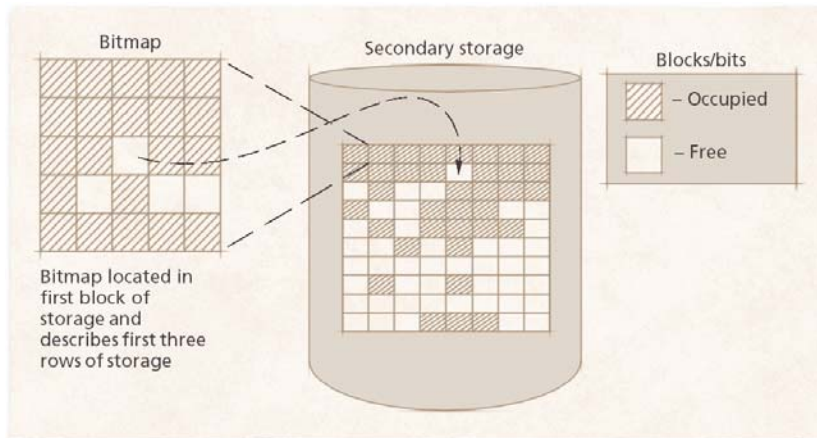
- A bitmap contains one bit for each block in memory
 - i th bit corresponds to the i th block on the storage device
- Advantage of bitmaps over free lists:
 - The file system can quickly determine if contiguous blocks are available at certain locations on secondary storage
- Disadvantage of bitmaps:
 - The file system may need to search the entire bitmap to find a free block, resulting in substantial execution overhead

© 2004 Deitel & Associates, Inc. All rights reserved.



13.7 Free Space Management

Figure 13.11 Free space management using a bitmap.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.8 File Access Control

- Files are often used to store sensitive data such as:
 - Credit card numbers
 - Passwords
 - Social security numbers
- Therefore, they should include mechanisms to control user access to data.
 - Access control matrix
 - Access control by user classes

© 2004 Deitel & Associates, Inc. All rights reserved.



13.8.1 Access Control Matrix

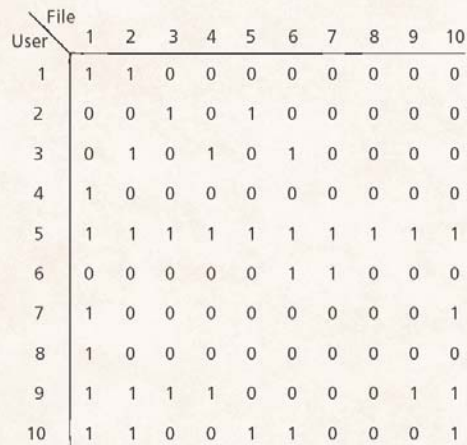
- Two-dimensional access control matrix:
 - Entry a_{ij} is 1 if user i is allowed access to file j
 - Otherwise $a_{ij} = 0$
- In an installation with a large number of users and a large number of files, this matrix generally would be large and sparse
- Inappropriate for most systems

© 2004 Deitel & Associates, Inc. All rights reserved.



13.8.1 Access Control Matrix

Figure 13.12 Access control matrix.



User \ File										
	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	0	0	0	0	0
2	0	0	1	0	1	0	0	0	0	0
3	0	1	0	1	0	1	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	1	1	0	0	0
7	1	0	0	0	0	0	0	0	0	1
8	1	0	0	0	0	0	0	0	0	0
9	1	1	1	1	0	0	0	0	1	1
10	1	1	0	0	1	1	0	0	0	1

© 2004 Deitel & Associates, Inc. All rights reserved.



13.8.2 Access Control by User Classes

- A technique that requires considerably less space is to control access to various user classes
- User classes can include:
 - The file owner
 - A specified user
 - Group
 - Project
 - Public
- Access control data
 - Can be stored as part of the file control block
 - Often consumes an insignificant amount of space

© 2004 Deitel & Associates, Inc. All rights reserved.



13.9 Data Access Techniques

- Today's operating systems generally provide many access methods
- Queued access methods
 - Used when the sequence in which records are to be processed can be anticipated, such as in sequential and indexed sequential accessing
 - Perform anticipatory buffering and scheduling of I/O operations
- Basic access methods
 - Normally used when the sequence in which records are to be accessed cannot be anticipated, particularly with direct accessing

© 2004 Deitel & Associates, Inc. All rights reserved.



13.9 Data Access Techniques

- Memory-mapped files
 - Map file data to a process's virtual address space instead of using a file system cache
 - Because references to memory-mapped files occur in a process's virtual address space, the virtual memory manager can make page-replacement decisions based on each process's reference pattern

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10 Data Integrity Protection

- Computer systems often store critical information, such as:
 - Inventories
 - Financial records
 - Personal information
- System crashes, natural disasters and malicious programs can destroy this information
- The results of such events can be catastrophic
- Operating systems and data storage systems should be fault tolerant:
 - Account for the possibility of disasters and provide techniques to recover from them

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.1 Backup and Recovery

- Backup techniques
 - Store redundant copies of information
- Recovery techniques
 - Enable the system to restore data after a system failure
- Physical safeguards such as locks and fire alarms are the lowest level of data protection
- Performing periodic backups is the most common technique used to ensure the continued availability of data

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.1 Backup and Recovery

- Physical backups
 - Duplicate a storage device's data at the bit level
- Logical backups
 - Store file system data and its logical structure
 - Inspect the directory structure to determine which files need to be backed up, then write these files to a backup device in a common, often compressed, archival format
- Incremental backups are logical backups that store only file system data that has changed since the previous backup

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- In systems that cannot tolerate loss of data or downtime, RAID and transaction logging are appropriate
- If a system failure occurs during a write operation, file data may be left in an inconsistent state
- Transaction-based file systems
 - Reduce data loss using atomic transactions:
 - Perform a group of operations in their entirety or not at all
 - If an error occurs that prevents a transaction from completing, it is rolled back by returning the system to the state before the transaction began

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- Transaction-based file systems
 - Reduce data loss using atomic transactions:
 - Perform a group of operations in their entirety or not at all
 - If an error occurs that prevents a transaction from completing, it is rolled back by returning the system to the state before the transaction began

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- Atomic transactions
 - Can be implemented by recording the result of each operation in a log file instead of modifying existing data
 - Once the transaction has completed, it is committed by recording a sentinel value in the log
- Checkpoints
 - To reduce the time spent reprocessing transactions in the log, most transaction-based systems maintain checkpoints that point to the last transaction that has been transferred to permanent storage
 - If the system crashes, it need only examine transactions after the checkpoint
- Shadow paging implements atomic transactions by writing modified data to a free block instead of the original block

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- Log-structured file systems (LFS)
 - Also called journaling file systems
 - Perform all file system operations as logged transactions to ensure that they do not leave the system in an inconsistent state
 - The entire disk serves as a log file
 - New data is written sequentially in the log file's free space

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

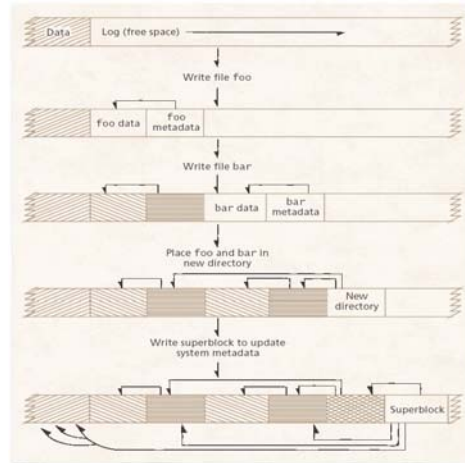
- Because modified directories and metadata are always written to the end of the log, an LFS might need to read the entire log to locate a particular file, leading to poor read performance
- To reduce this problem, an LFS
 - Caches locations of file system metadata
 - Occasionally writes inode maps or superblocks to the log to indicate the location of other metadata
 - Enables the operating system to locate and cache file metadata quickly when the system boots

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

Figure 13.13 Log-structured file system.



© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- Some file systems attempt to reduce the cost of log-structured file systems by using a log only to store metadata
 - Ensures file system integrity with relatively low overhead
 - Does not ensure file integrity in the event of a system failure

© 2004 Deitel & Associates, Inc. All rights reserved.



13.10.2 Data Integrity and Log-Structured File Systems

- Data is written to the log sequentially
 - Each LFS write requires only a single seek while there is still space on disk
- When the log fills, the file system's free space is likely to be fragmented
- This can lead to poor read and write performance
- To address this issue, an LFS can create contiguous free space in the log by copying valid data to a contiguous region at the end of the log

© 2004 Deitel & Associates, Inc. All rights reserved.



13.11 File Servers and Distributed Systems

- One approach to handling nonlocal file references in a computer network is to route all such requests to a file server
 - A computer system dedicated to resolving intercomputer file references
 - Centralizes control of these references
 - File server could easily become a bottleneck, because all client computers send all requests to the server
- A better approach is to let the separate computers communicate directly with one another

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12 Database Systems

- Database
 - Centrally controlled collection of data stored in a standardized format
- A database system involves:
 - Data
 - Hardware on which the data resides
 - Software that controls access to data (called a database management system or DBMS)

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.1 Advantages of Database Systems

- Databases reduce data redundancy and prevent data from being in an inconsistent state
 - Redundancy is reduced by combining identical data from separate files
- Databases also facilitate data sharing

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.2 Data Access

- Data independence
 - Applications need not be concerned with how data is physically stored or accessed
 - Makes it possible for the storage structure and accessing strategy to be modified in response to the installation's changing requirements, but without the need to modify functioning applications



13.12.2 Data Access

- Database languages
 - Allow database independence by providing a standard way to access information
 - A database language consists of:
 - Data definition language (DDL)
 - Data manipulation language (DML)
 - Query language



13.12.2 Data Access

- Database languages (cont.)
 - A DDL specifies how data are organized and related and the DML enables data to be modified
 - A query language is a part of the DML that allows users to create queries that search the database for data that meets certain criteria
 - The Structured Query Language (SQL) is currently one of the most popular database languages

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.2 Data Access

- Distributed database
 - Database that is spread throughout the computer systems of a network
 - Facilitates efficient data access across many sets of data that reside on different computers

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.3 Relational Database Model

- Databases are based on models that describe how data and their relationships are viewed
- Relational model is a logical structure rather than a physical one
- The principles of relational database management are independent of the physical implementation of data structures

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.3 Relational Database Model

- Relations
 - Indicate the various attributes of an entity
 - Any particular element of a relation is called a tuple (row).
 - Each attribute (column) of the relation belongs to a single domain
 - The number of attributes in a relation is the degree of the relation
 - A projection operation forms a subset of the attributes
 - A join operation combines relations to produce more complex relations
- The relational database model is relatively easy to implement

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.3 Relational Database Model

Figure 13.14 Relation in a relational database.

Relation: EMPLOYEE

	Number	Name	Department	Salary	Location
	23603	Jones, A.	413	1100	New Jersey
	24568	Kerwin, R.	413	2000	New Jersey
A tuple {	34589	Larson, P.	642	1800	Los Angeles
	35761	Myers, B.	611	1400	Orlando
	47132	Neumann, C.	413	9000	New Jersey
	78321	Stevens, T.	611	8500	Orlando

Primary key An attribute

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.3 Relational Database Model

Figure 13.15 Relation formed by projection.

Relation: DEPARTMENT-LOCATOR

<i>Department</i>	<i>location</i>
413	NEW JERSEY
611	ORLANDO
642	LOS ANGELES

© 2004 Deitel & Associates, Inc. All rights reserved.



13.12.3 Relational Database Model

Figure 13.16 SQL query.

```
1  -- SQL query to generate the table in Fig. 13.15
2  SELECT DISTINCT Department, Location
3  FROM EMPLOYEE
4  ORDER BY Department ASC
5
```



13.12.4 Operating Systems and Database Systems

- Various operating system services support database management systems, namely:
 - Buffer pool management
 - File system
 - Scheduling
 - Process management
 - Interprocess communication
 - Consistency control
 - Paged virtual memory
- Most of these features are not specifically optimized to DBMS environments, so DBMS designers have tended to bypass operating system services in favor of supplying their own

