Understanding Operating Systems
Sixth Edition

Chapter 8
File Management

Learning Objectives

After completing this chapter, you should 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

Learning Objectives (cont'd.)

- The advantages and disadvantages of contiguous, noncontiguous, and indexed file storage techniques
- Comparisons of sequential and direct file access
- Access control techniques and how they compare
- The role of data compression in file storage

The File Manager

- **File management system**
  - Software
- File access responsibilities
  - Creating, deleting, modifying, controlling
- Support for libraries of programs
  - Online users
    - Spooling operations
    - Interactive computing
- Collaborates with device manager

Responsibilities of the File Manager

- **Four tasks**
  - File storage tracking
  - Policy implementation
    - Determine where and how files are stored
    - Efficiently use available storage space
    - Provide efficient file access
  - File allocation if user access cleared
    - Record file use
  - File deallocation
    - File returned to storage
    - Communicate file availability

Responsibilities of the File Manager (cont'd.)

- Policy determines:
  - File storage location
  - System and user access
    - Uses device-independent commands
- Access to material (who and how)
  - Two factors
- Flexibility of access to information (Factor 1)
  - Shared files
  - Providing distributed access
  - Allowing users to browse public directories

Responsibilities of the File Manager (cont'd.)

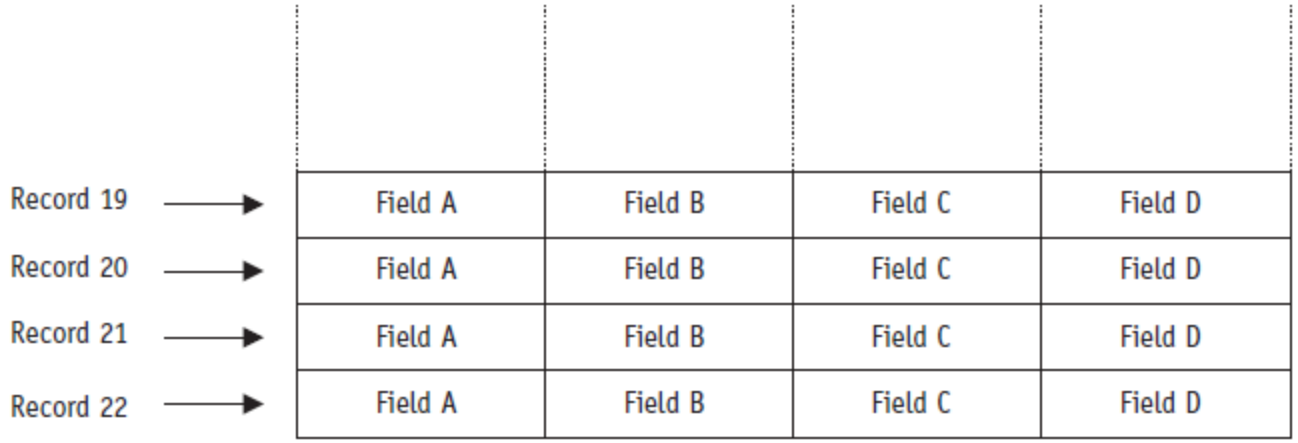
- Subsequent protection (Factor 2)
  - Prevent system malfunctions
  - Security checks
    - Account numbers and passwords
- **File allocation**
  - Activate secondary storage device, load file into memory, update records
- **File deallocation**
  - Update file tables, rewrite file (if revised), notify waiting processes of file availability

Definitions

- **Field**
  - Group of related bytes
  - Identified by user (name, type, size)
- **Record**
  - Group of related fields

(figure 8.1)

Files are made up of records. Records consist of fields.



Definitions (cont'd.)

- **File**
  - Group of related records
  - Information used by specific application programs
    - Report generation
  - Flat file
    - No connections to other files, no dimensionality
- **Databases**
  - Groups of related files
  - Interconnected at various levels
    - Give users flexibility of access to stored data
- **Program files**
  - Contain instructions
- **Directories**
  - Listings of filenames and their attributes

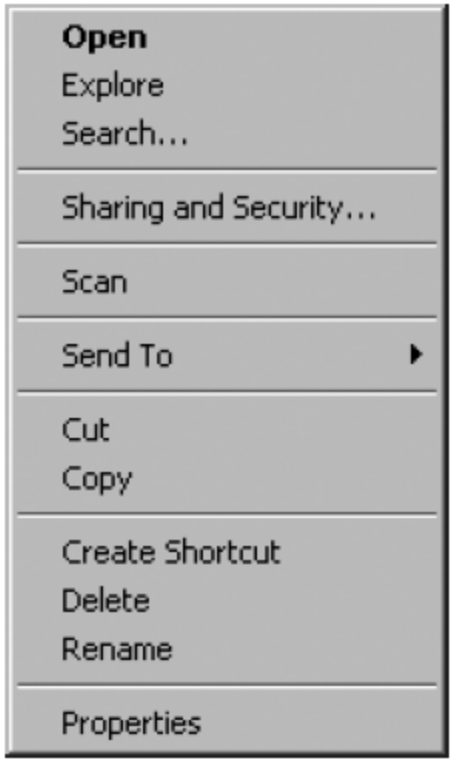
Interacting with the File Manager

- User Commands
  - OPEN, DELETE, RENAME, COPY
- **Device independent**
  - Physical location knowledge not needed
    - Cylinder, surface, sector
  - Device medium knowledge not needed
    - Tape, magnetic disk, optical disc, flash storage
  - Network knowledge not needed

Interacting with the File Manager (cont'd.)

(figure 8.2)

Typical menu of file options.



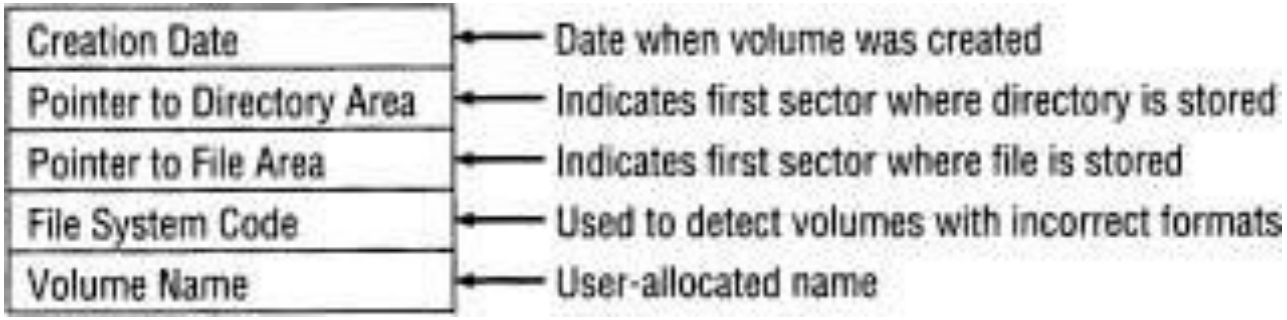
Interacting with the File Manager (cont'd.)

- Logical commands
  - Broken into lower-level signals
  - Example: READ
    - Move read/write heads to record cylinder
    - Wait for rotational delay (sector containing record passes under read/write head)
    - Activate appropriate read/write head and read record
    - Transfer record to main memory
    - Send flag indicating free device for another request
- Performs error checking and correction
  - No need for error-checking code in programs

Typical Volume Configuration

- **Volume**
  - Secondary storage unit (removable, nonremovable)
  - Multifile volume
    - Contains many files
  - Multivolume files
    - Extremely large files spread across several volumes
- Volume name
  - File manager manages
  - Easily accessible
    - Innermost part of CD, beginning of tape, first sector of outermost track

Typical Volume Configuration (cont'd.)



(figure 8.3)

The volume descriptor, which is stored at the beginning of each volume, includes this vital information about the storage unit.

Typical Volume Configuration (cont'd.)

- **Master file directory (MFD)**
- Stored immediately after volume descriptor
- Lists
  - Names and characteristics of every file in volume
    - File names (program files, data files, system files)
  - Subdirectories
    - If supported by file manager
- Remainder of volume is used for file storage

Typical Volume Configuration (cont'd.)

- Single directory per volume
  - Supported by early operating systems
- Disadvantages
  - Long search time for individual file
  - Directory space filled before disk storage space filled
  - Users cannot create subdirectories
  - Users cannot safeguard their files
  - Each program needs unique name
    - Even those serving many users

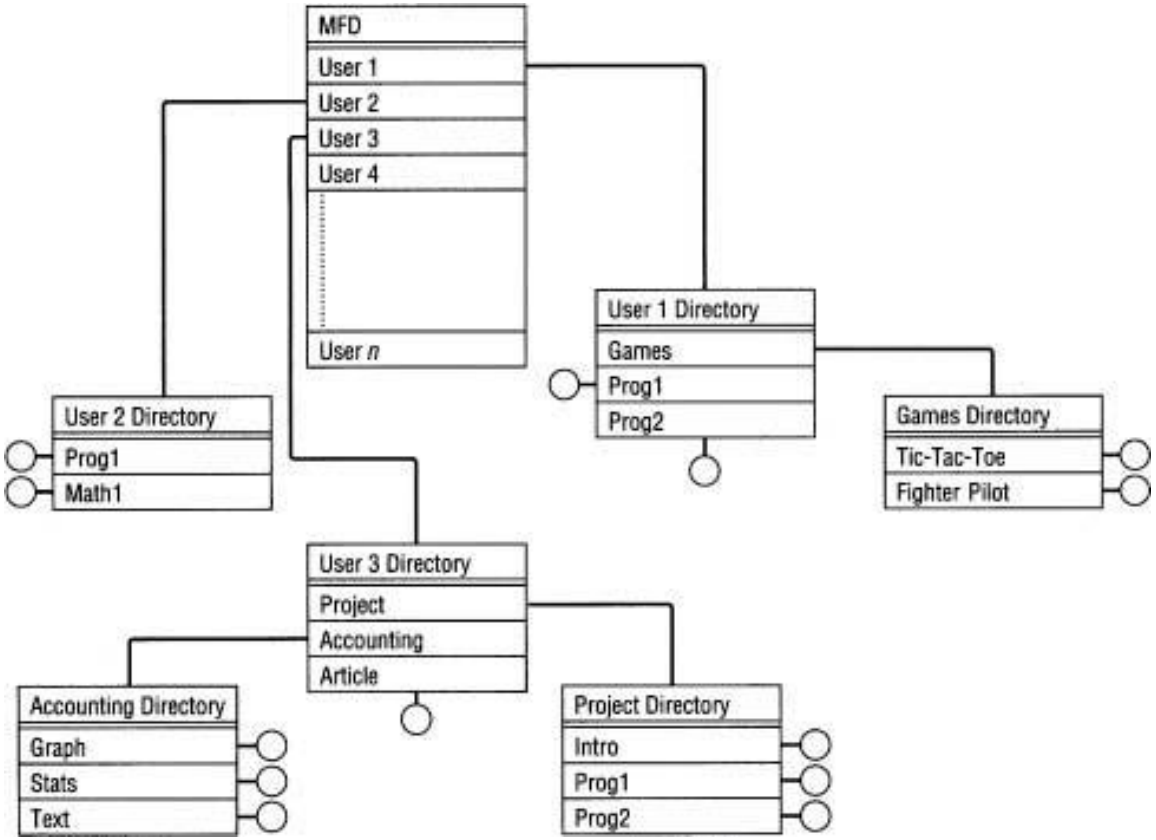
Introducing Subdirectories

- **File managers**
  - Create MFD for each volume
    - Contains file and subdirectory entries
  - Improvement over single directory scheme
    - Problems remain: unable to logically group files
- **Subdirectory**
  - Created upon account opening
  - Treated as file
    - Flagged in MFD as subdirectory
    - Unique properties

Introducing Subdirectories (cont'd.)

- File managers today
  - Users create own subdirectories (folders)
    - Related files grouped together
  - Implemented as upside-down tree
    - Efficient system searching of individual directories
    - May require several directories to reach file

Introducing Subdirectories (cont'd.)



(figure 8.4)

File directory tree structure. The "root" is the MFD shown at the top, each node is a directory file, and each branch is a directory entry pointing to either another directory or to a real file. All program and data files subsequently added to the tree are the leaves, represented by circles.

Introducing Subdirectories (cont'd.)

- **File descriptor**
  - Filename: ASCII code
  - File type: organization and usage
    - System dependent
  - File size: for convenience
  - File location
    - First physical block identification
  - Date and time of creation
  - Owner
  - Protection information: access restrictions
  - Record size: fixed size, maximum size

File-Naming Conventions

- Filename components
  - Relative filename and extension
- **Complete filename (absolute filename)**
  - Includes all path information
- **Relative filename**
  - Name without path information
  - Appears in directory listings, folders
  - Provides filename differentiation within directory
  - Varies in length
    - One to many characters
    - Operating system specific

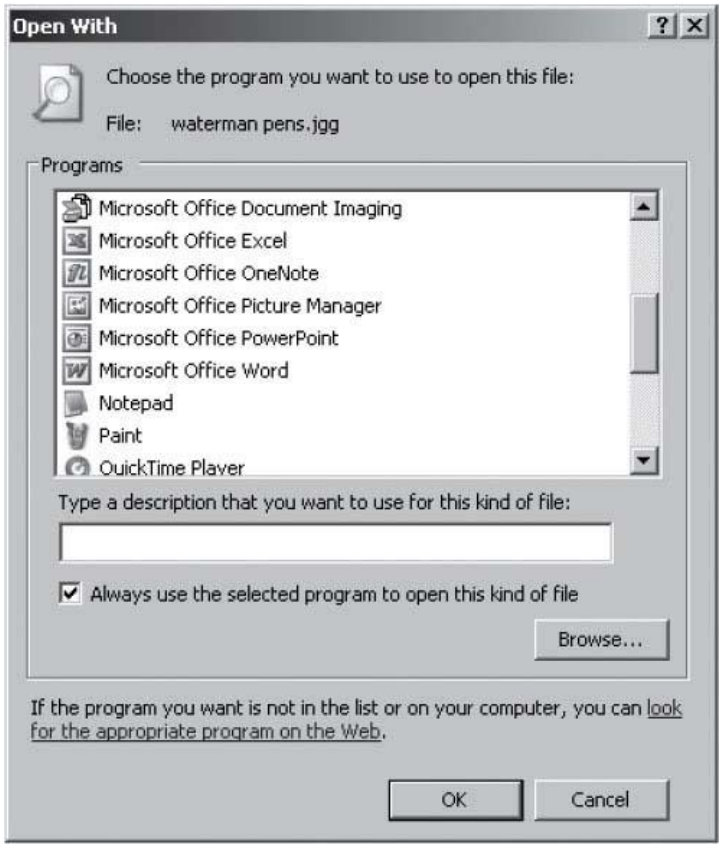
File-Naming Conventions (cont'd.)

- **Extensions**
  - Appended to relative filename
    - Two to three characters
    - Separated by period
    - Identifies file type or contents
  - Example
    - BASIA_TUNE.MP3
  - Unknown extension
    - Requires user intervention

File-Naming Conventions (cont'd.)

(figure 8.5)

To open a file with an unrecognized extension, Windows asks the user to choose an application to associate with that type of file.



File-Naming Conventions (cont'd.)

- Operating system specifics
  - Windows
    - Drive label and directory name, relative name, and extension

    C:\aesonmez\documents\document.doc
  - UNIX/Linux
    - Forward slash (root), first subdirectory, sub-subdirectory, file's relative name

    /usr/aesonmez/documents/document.doc

File Organization

- Arrangement of records within files
- All files composed of records
- Modify command
  - Request to access record within a file

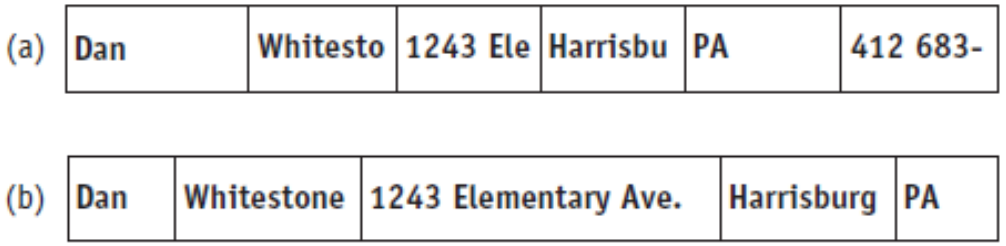
Record Format

- **Fixed-length records**
  - Direct access: easy
  - Record size critical
  - Ideal for data files
- **Variable-length records**
  - Direct access: difficult
  - No empty storage space and no character truncation
  - File descriptor stores record format
  - Used with files accessed sequentially
    - Text files, program files
  - Used with files using index to access records

Record Format (cont'd.)

(figure 8.6)

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 longer to access it.



Physical File Organization

- Describes:
  - Record arrangement
  - Medium characteristics
- Magnetic disks file organization
  - Sequential, direct, indexed sequential
- File organization scheme selection considerations
  - Volatility of data
  - Activity of file
  - Size of file
  - Response time

Physical File Organization (cont'd.)

- **Sequential record organization**
  - Records stored and retrieved serially
    - One after the other
  - Easiest to implement
  - File search: beginning until record found
  - Optimization features may be built into system
    - Select key field from record and sort before storage
    - Complicates maintenance algorithms
    - Preserve original order when records added, deleted

Physical File Organization (cont'd.)

- **Direct record organization**
  - Uses direct access files
  - Direct access storage device implementation
    - Random organization
    - Random access files
  - **Relative address** record identification
    - Known as **logical addresses**
    - Computed when records stored, retrieved
  - Uses **hashing algorithms** to transform a **key field**

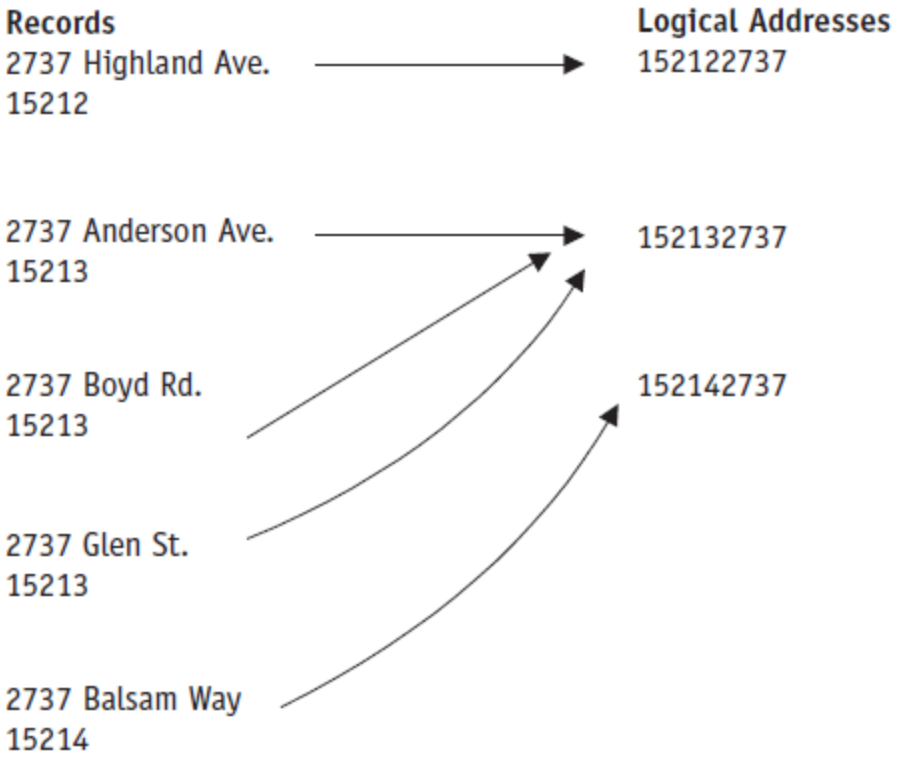
Physical File Organization (cont'd.)

- Direct record organization (cont'd.)
  - Advantages
    - Fast record access
    - Sequential access if starting at first relative address and incrementing to next record
    - Updated more quickly than sequential files
    - No preservation of records order
    - Adding, deleting records is quick
  - Disadvantages
    - Hashing algorithm collision
    - Similar keys

Physical File Organization (cont'd.)

(figure 8.7)

The hashing algorithm causes a collision. Using a combination of street address and postal code, it generates the same logical address (152132737) for three different records.



Physical File Organization (cont'd.)

- **Indexed sequential record organization**
  - Combines the best of sequential and direct access
  - Indexed Sequential Access Method software: creates, maintains
  - Advantage: no collisions (no hashing algorithm)
    - Generates index file for record retrieval
    - Divides ordered sequential file into equal sized blocks
    - Each entry in index file contains the highest record key and physical data block location
    - Search index file

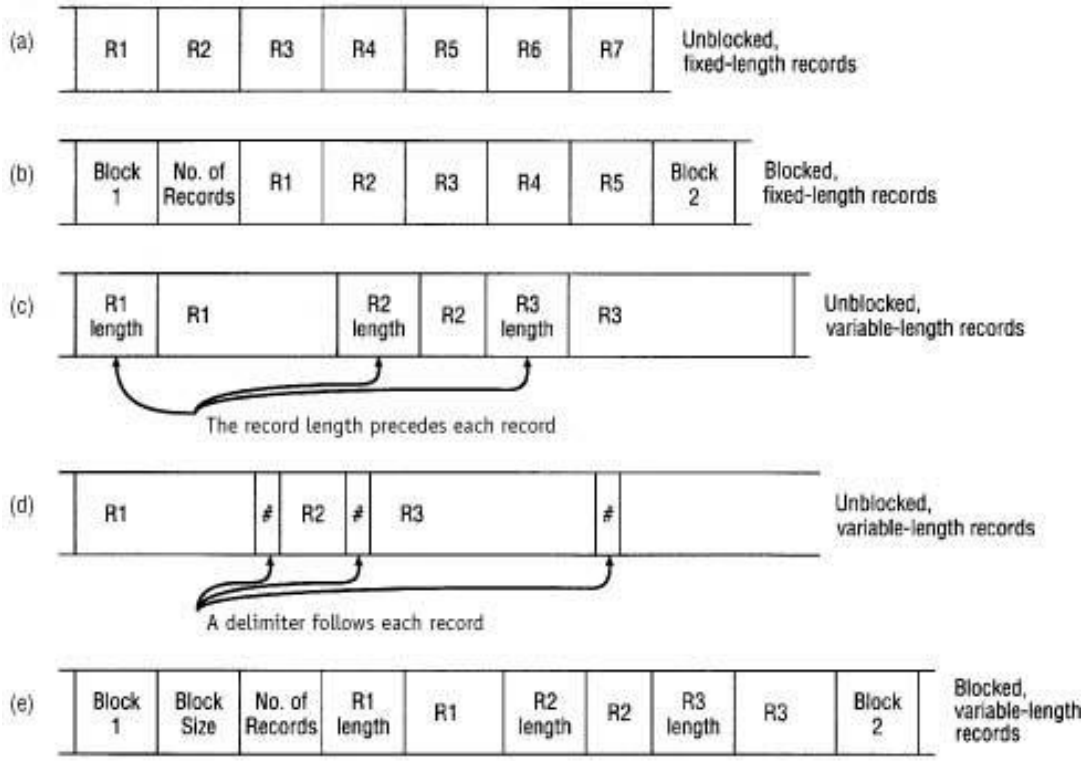
Physical Storage Allocation

- File manager works with files
  - As whole units
  - As logical units or records
- Within file
  - Records must have same format
  - Record length may vary
- Records subdivided into fields
- Application programs manage record structure
- File storage
  - Refers to record storage

Physical Storage Allocation (cont'd.)

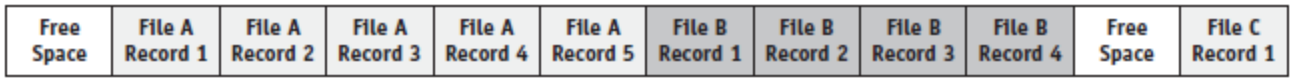
(figure 8.8)

Every record in a file must have the same format but can be of different sizes, as shown in these five examples of the most common record formats. The supplementary information in (b), (c), (d), and (e) is provided by the File Manager, when the record is stored.



Contiguous Storage

- Records stored one after another
  - Advantages
    - Any record found once starting address, size known
    - Easy direct access
  - Disadvantages
    - Difficult file expansion, fragmentation



(figure 8.9)

With contiguous file storage, File A can't be expanded without being rewritten to a larger storage area. File B can be expanded, by only one record replacing the free space preceding File C.

Noncontiguous Storage

- Files use any available disk storage space
- File records stored in contiguous manner
  - If enough empty space
- Remaining file records and additions
  - Stored in other disk sections (extents)
  - Extents
    - Linked together with pointers
    - Physical size determined by operating system
    - Usually 256 bytes

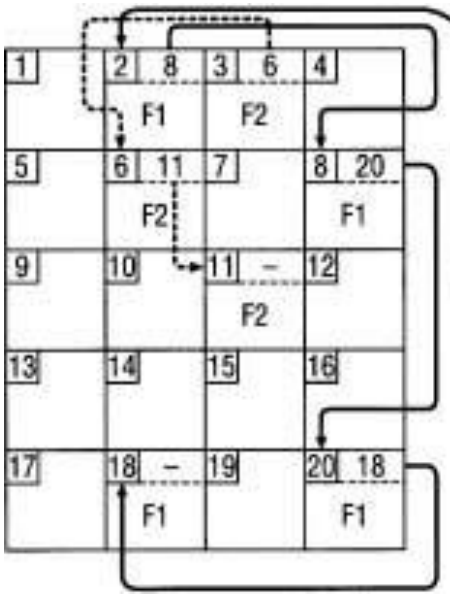
Noncontiguous Storage (cont'd.)

- File extents linked in two ways
  - Storage level
    - Each extent points to next one in sequence
    - Directory entry
    - Filename, storage location of first extent, location of last extent, total number of extents (not counting first)
  - Directory level
    - Each extent listed with physical address, size, pointer to next extent
    - Null pointer indicates last one

Noncontiguous Storage (cont'd.)

(figure 8.10)

Noncontiguous file storage with linking taking place at the storage level. File 1 starts in address 2 and continues in addresses 8, 20, and 18. The directory lists the file's starting address, ending address, and the number of extents it uses. Each block of storage includes its address and a pointer to the next block for the file, as well as the data itself.



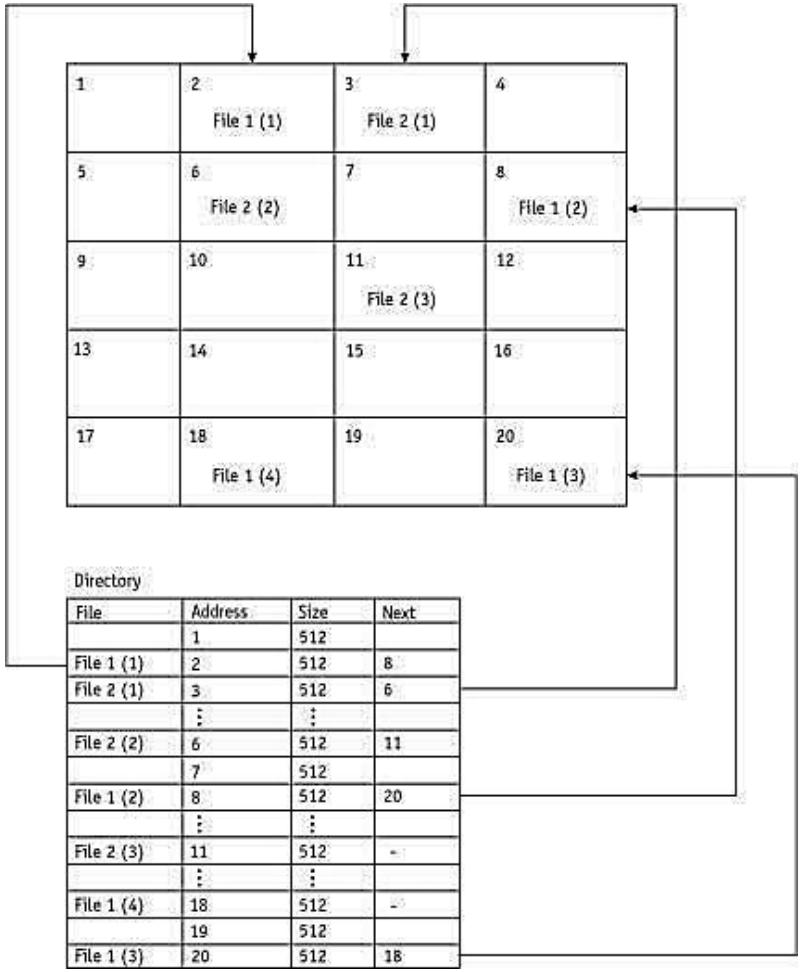
Directory

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

Noncontiguous Storage (cont'd.)

(figure 8.11)

Noncontiguous storage allocation with linking taking place at the directory level for the files shown in Figure 8.10.



Noncontiguous Storage (cont'd.)

- Advantage
  - Eliminates external storage fragmentation
  - Eliminates need for compaction
- Disadvantage
  - No direct access support
    - Cannot determine specific record’s exact location

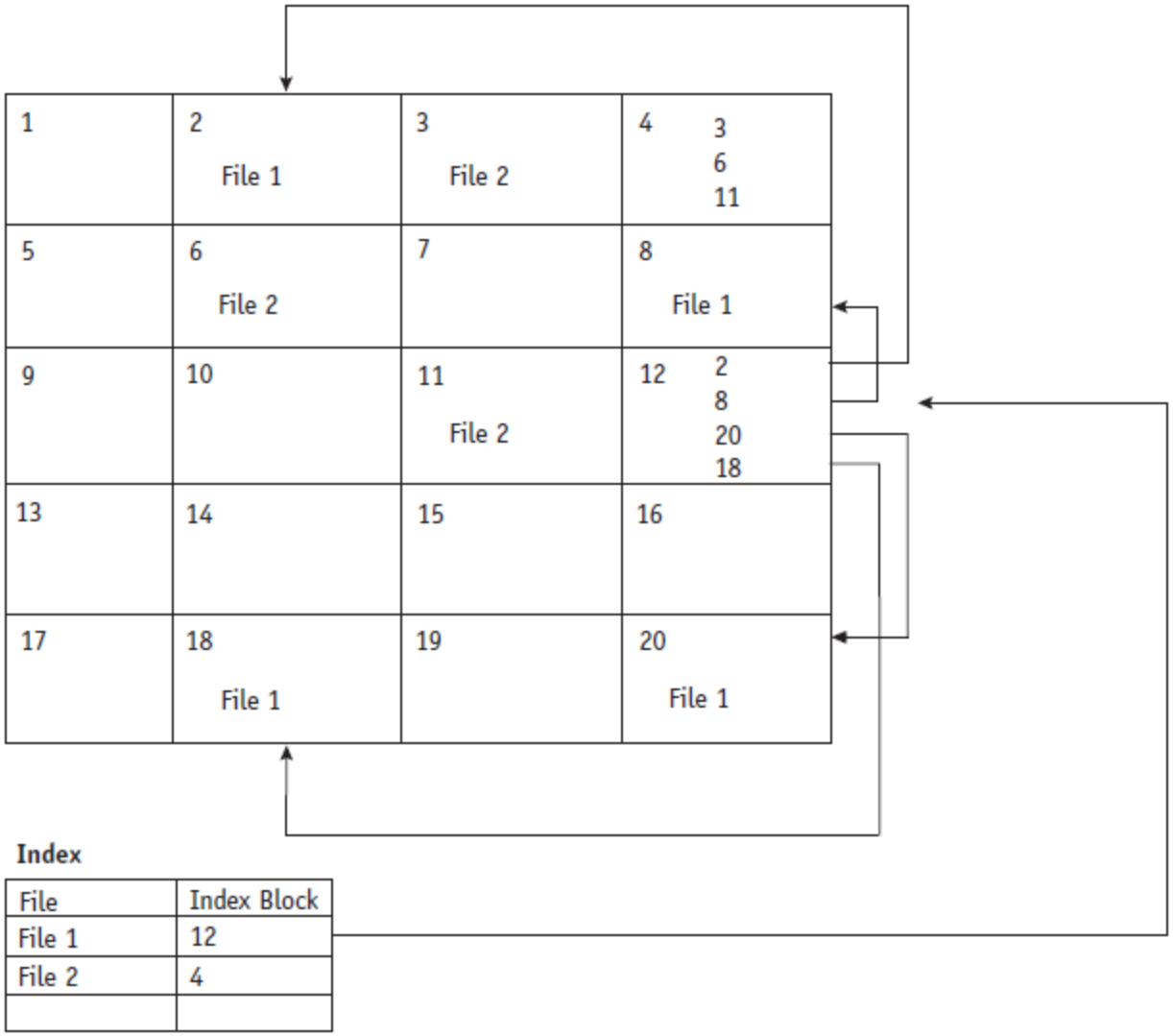
Indexed Storage

- Allows direct record access
  - Brings pointers together
    - Links every extent file into index block
- Every file has own index block
  - Disk sector addresses for file
  - Lists entry in order sectors linked
- Supports sequential and direct access
- Does not necessarily improve storage space use
- Larger files experience several index levels

Indexed Storage (cont'd.)

(figure 8.12)

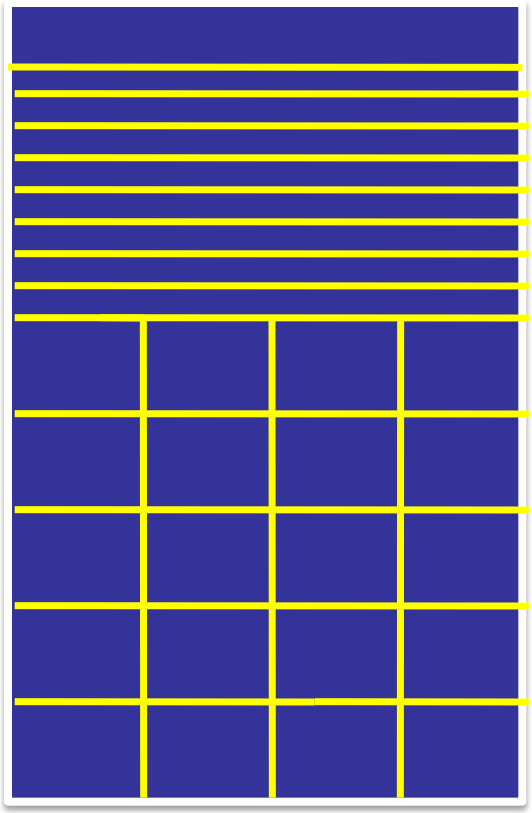
Indexed storage allocation with a one-level index, allowing direct access to each record for the files shown in Figures 8.10 and 8.11.



UNIX Version 7 Implementation

- Each disk partition contains:
  - a ***superblock*** containing the parameters of the file system disk partition
  - an ***i-list*** with one ***i-node*** for each file or directory in the disk partition and a ***free list***.
  - the ***data blocks*** (512 bytes)

A disk partition (“filesystem”)



Superblock

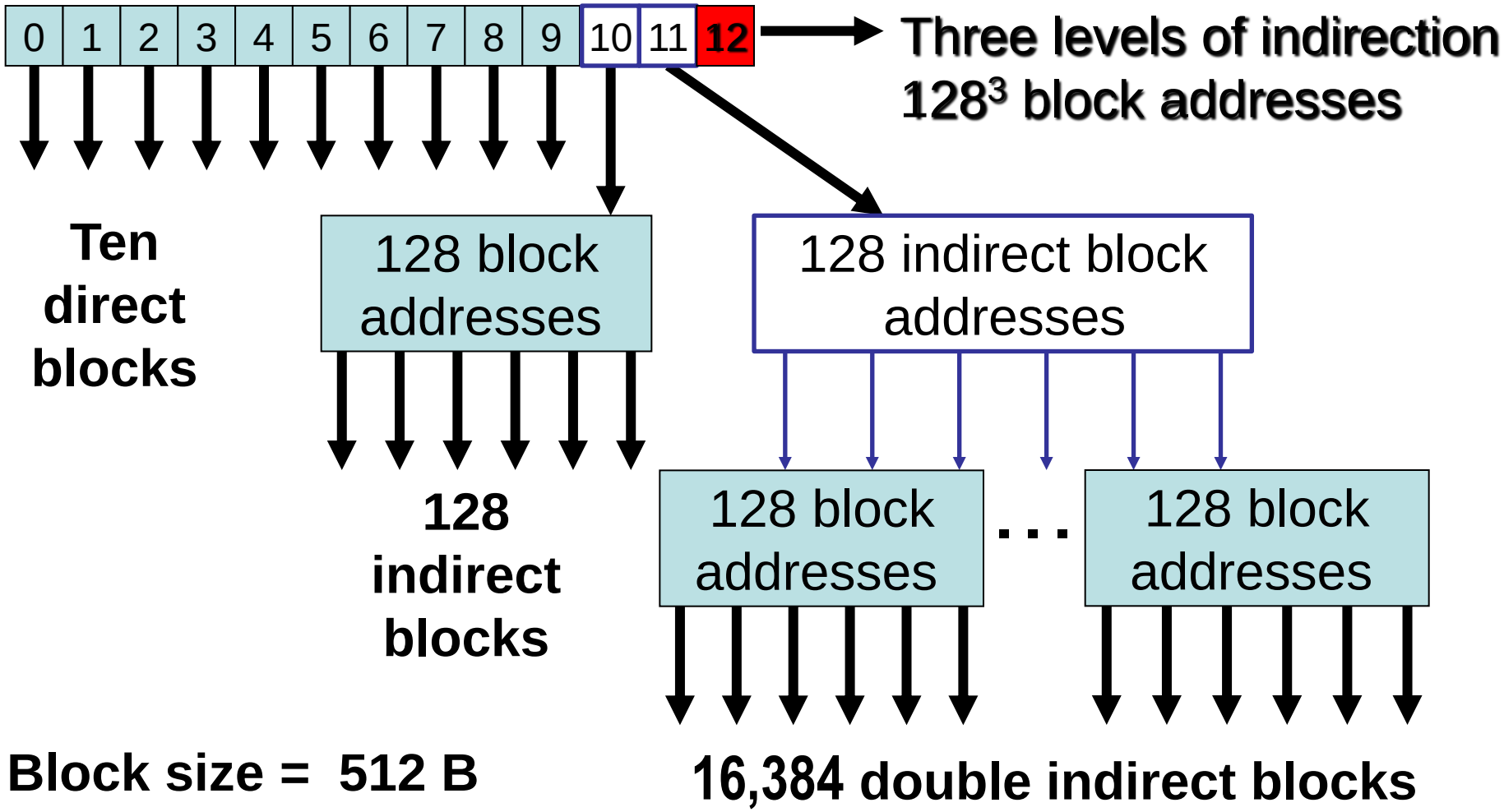
I-nodes

Data Blocks

The i-node

- Each ***i-node*** contains:
  - The ***user-id*** and the ***group-id*** of the file owner
  - The file protection bits,
  - The file size,
  - The times of file creation, last usage and last modification,
  - The *number* of directory entries pointing to the file, and
  - A flag indicating if the file is a directory, an ordinary file, or a special file.
  - Thirteen block addresses
- The file name(s) can be found in the directory entries pointing to the i-node

Storing block addresses



How it works (I)

- First ten blocks of file can be accessed directly from i-node
  - 10x512= 5,120 bytes
- Indirect block contains 512/4 = 128 addresses
  - 128x512= 64 kilobytes
- With two levels of indirection we can access 128x128 = 16K blocks
  - 16Kx512 = 8 megabytes

How it works (II)

- With three levels of indirection we can access 128x128X128 = 2M blocks
  - 2Mx512 = 1 gigabyte
- Maximum file size is

  1 GB + 8 MB + 64KB + 5KB

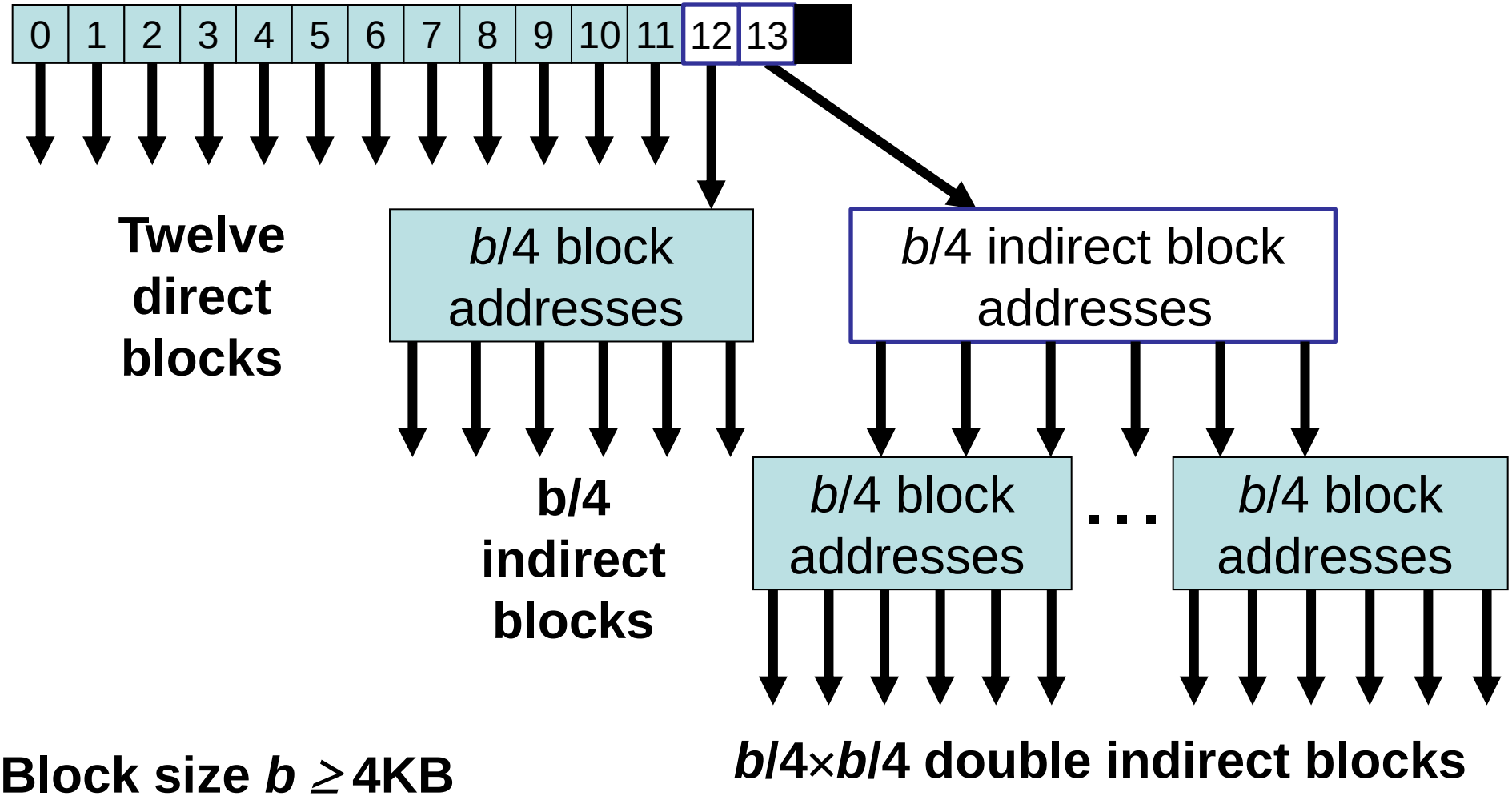
FFS Modifications

- BSD (Berkeley Software Distribution) introduced the “fast file system” (FFS)
  - ***Superblock*** is replicated on different cylinders of disk
  - Disk is divided into ***cylinder groups***
  - Each cylinder group has its own i-node table
    - It minimizes disk arm motions

The FFS i-node

- I-node has now 15 block addresses
- Minimum block size is 4K
  - 15^{th} block address is never used

FFS organization (I)



Block size $b \geq 4$KB

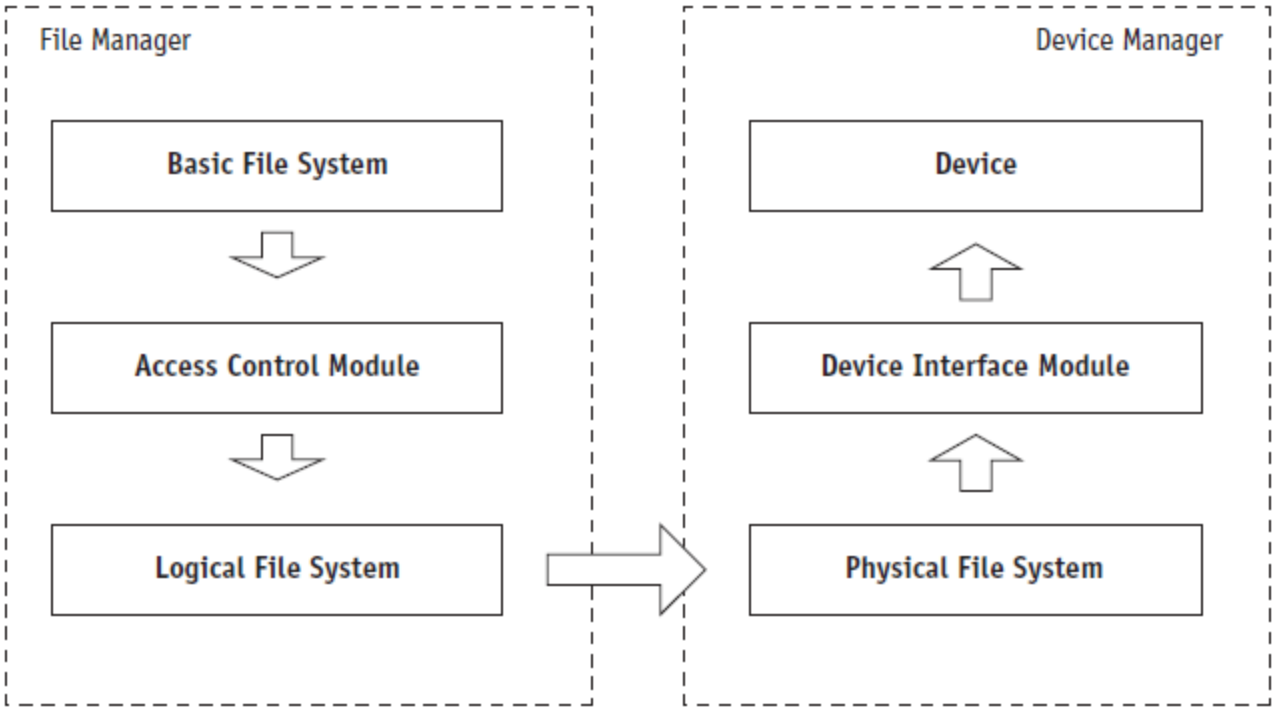
How it works

- In a 32 bit architecture, file size is limited to 2^{32} bytes, that is, 4GB
- When block size is 4KB, we can access
  - **12 ×4KB = 48 KB *directly*** from i-node
  - **1,024 ×4KB = 4 MB** with ***one level*** of indirection
  - **4GB – 48KB – 4MB** with ***two levels*** of indirection

Windows NTFS data structures

- ***Master File Table*** (MFT)
  - Contains most metadata
  - Equivalent to UNIX i-node table
- Each file can have one or more MFT records depending on file size and attribute complexity
- MFT records contain
  - Pointers to data blocks for most files
  - Contents of very small files

Levels in a File Management System



(figure 8.14)

Typical modules of a file management system showing how information is passed from the File Manager to the Device Manager.

Levels in a File Management System (cont'd.)

- Level implementation
  - Structured and modular programming techniques
  - Hierarchical
    - Highest module passes information to lower module
- Modules further subdivided
  - More specific tasks
- Uses information of basic file system
  - Logical file system transforms record number to byte address

Levels in a File Management System (cont'd.)

- **Verification** at every level
  - Directory level
    - File system checks if requested file exists
  - Access control verification module
    - Determines whether access allowed
  - Logical file system
    - Checks if requested byte address within file limits
  - Device interface module
    - Checks if storage device exists

Access Control Verification Module

- File sharing
  - Data files, user-owned program files, system files
  - Advantages
    - Save space, synchronized updates, resource efficiency
  - Disadvantage
    - Need to protect file integrity
  - Five possible file actions
    - READ only, WRITE only, EXECUTE only, DELETE only, combination
  - Four methods

Access Control Matrix

- Advantages
  - Easy to implement
  - Works well in system with few files, users

(table 8.2)

The access control matrix showing access rights for each user for each file. User 1 is allowed unlimited access to File 1 but is allowed only to read and execute File 4 and is denied access to the three other files. R = Read Access, W = Write Access, E = Execute Access, D = Delete Access, and a dash (-) = Access Not Allowed.

	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

Access Control Matrix (cont'd.)

- Disadvantages
  - As files and user increase, matrix increases
    - Possibly beyond main memory capacity
  - Wasted space: due to null entries

(table 8.3)

The five access codes for User 2 from Table 8.2. The resulting code for each file is created by assigning a 1 for each checkmark, and a 0 for each blank space.

Access	R	W	E	D	Resulting Code
R-E-	√		√		1010
R-E-	√		√		1010
RWED	√	√	√	√	1111
----					0000
----					0000

Access Control Lists

- Modification of access control matrix technique

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 8.4)

An access control list showing which users are allowed to access each file. This method requires less storage space than an access control matrix.

Access control lists (II)

Access Control Lists (cont'd.)

- Contains user names granted file access
  - User denied access grouped under “WORLD”
- Shorten list by categorizing users
  - 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

Access control lists (III)

- ***Main advantage:***
  - ***Very flexible:*** can easily add new users or change/revoke permissions of existing users
- ***Main disadvantages:***
  - ***Very slow:*** must authenticate user at each access
  - Can take more space than the file itself

Capability Lists (Tickets)

- Lists every user and files each has access to
- Can control access to devices as well as to files
- Most common

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 8.5)

A capability list shows files for each user and requires less storage space than an access control matrix; and when users are added or deleted from the system, is easier to maintain than an access control list.

Tickets (II)

Tickets (III)

- ***Main advantage:***
  - ***Very fast:*** must only check that the ticket is valid
- ***Main disadvantages:***
  - ***Less flexible than access control lists:*** cannot revoke individual tickets
  - Ticket holders can make copies of tickets and distribute them to other users

Conclusion

- Best solution is to combine both approaches
  - Use access control lists for long-term management of permissions
    - Once a user has been authenticated, give him or her a ticket
    - Limit ticket lifetimes to force users to be authenticated from time to time

UNIX solution

- UNIX
  - Checks access control list of file whenever a file is opened
  - Lets file descriptor act as a ticket until the file is closed

UNIX access control lists (I)

- File owner can specify three access rights
  - read
  - write
  - execute

  for
  - herself (user)
  - a group in /etc/group (group)
  - all other users (other)

UNIX access control lists (II)

- **`rwx------`**

  Owner can do everything she wants with her file and nobody else can access it
- **`rw-r--r--`**

  Owner can read from and write to the file, everybody else can read the file
- **`rw-rw----`**

  Owner and member of group can read from and write to the file

Data Compression

- A technique used to save space in files
- Text decompression
- Other decompression schemes

Text Compression

- **Records with repeated characters**
  - Repeated characters are replaced with a code
    - ADAMSbbbbbbbbb can be represented as ADAMSb10
    - 300000000 can be represented as 3#8
- **Repeated terms**
  - Compressed using symbols to represent most commonly used words
  - University student database common words
    - Student, course, grade, department each be represented with single character
- **Front-end compression**
  - Entry takes given number of characters from previous entry that they have in common

Repeated terms

- Compressed using symbols to represent most commonly used words
- University student database common words
  - Student, course, grade, department each be represented with single character

Original List	Compressed List
Smith, Betty	Smith, Betty
Smith, Donald	7Donald
Smith, Gino	7Gino
Smithberger, John	5berger, John
Smithbren, Ali	6ren, Ali
Smithco, Rachel	5co, Rachel
Smithers, Renny	7s, Renny

Other Compression Schemes

- Large files
  - Video and music
    - ISO MPEG standards
  - Photographs
- ISO
  - International Organization for Standardization

Summary

- File manager
  - Controls every file and processes user commands
  - Manages access control procedures
    - Maintain file integrity and security
  - File organizations
    - Sequential, direct, indexed sequential
  - Physical storage allocation schemes
    - Contiguous, noncontiguous, indexed
  - Record types
    - Fixed-length versus variable-length records
  - Four access methods

- **2.** A 32-bit Berkeley UNIX file system has a block size of 16 kilobytes. How many blocks of a given file can be accessed :

(a) using the block addresses stored in the i-node?

(b) with one level of indirection?

(c) with two levels of indirection?

(A) 12 blocks

(B) 16K/4 =4096 blocks

(C) 256K -4096 -12 blocks

- Since we have a 32 bit file system, the maximum file size is 4 Gigabytes. The maximum number of 16 KB blocks in a file is thus 4GB/16KB = 2^32/2^14 = 2^18=
- 256K.