目录 Contents

File Management

9.1 File and File System

1 File and File System

- Why we need files?

- **File**: Data collections created by users
- Desirable properties of files:
  - Long-term existence: files are stored on disk or other secondary storage and do not disappear when a user logs off
  - Sharable between processes: files have names and can have associated access permissions that permit controlled sharing
  - Structure: files can be organized into hierarchical or more complex structure to reflect the relationships among files

File Systems

- **File System**: Provide a **means** to store data organized as files as well as a collection of functions that can be performed on files
- The File System is one of the most important parts of the OS to a user
- Typical operations include:
  - Create
  - Delete
  - **Open**
  - Close
  - Read
  - Write

Terms About Files

- **Field**: basic element of data, contains a single value
  - fixed or variable length
- **Record**: collection of related fields that can be treated as a unit by some application program
  - fixed or variable length
- **File**: collection of similar records, treated as a single entity
  - may be referenced by name
  - access control restrictions usually apply at the file level
- **Database**: collection of related data
  - relationships among elements of data are explicit
  - designed for use by a number of different applications
  - consists of one or more types of files

File Management System Objectives

- Meet the data management **needs** of the user
- Guarantee that the data in the file are valid
- Optimize performance
- Provide I/O support for a variety of storage device types
- Minimize the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines to user processes
- Provide I/O support for multiple users in the case of multiple-user systems

Minimal User Requirements

- should be able to create, delete, read, write and modify files
- may have controlled access to other users' files
- may control what type of accesses are allowed to the files
- should be able to restructure the files in a form appropriate to the problem
- should be able to move data between files
- should be able to back up and recover files in case of damage
- should be able to access his or her files by name rather than by numeric identifier

9.1 File and File System

2 File System Software Architecture

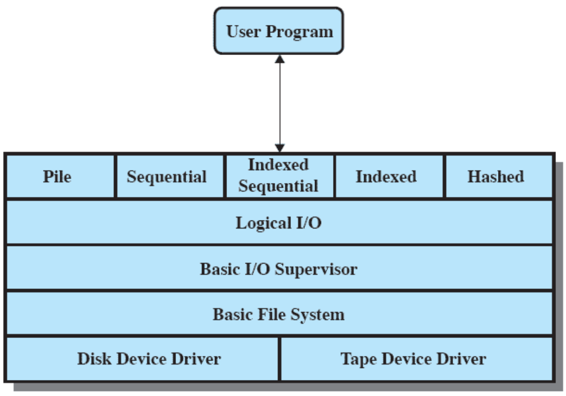


Figure 12.1 File System Software Architecture

Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- Responsible for **starting I/O operations** on a device
- Processes the **completion of an I/O request**

> Considered to be part of the operating system

Basic File System

- Also referred to as **the physical I/O level**
- Primary interface with the environment outside the computer system
- Deals with **blocks of data** that are exchanged with disk or tape systems
- Concerned with **the placement of blocks** on the secondary storage device
- Concerned with **buffering blocks** in main memory

> Considered to be part of the operating system

Basic I/O Supervisor

- Responsible for all file I/O initiation and termination
- Control structures that deal with device I/O, scheduling, and file status are maintained
- Selects the device on which I/O is to be performed
- Concerned with scheduling disk and tape accesses to optimize performance
- I/O buffers are assigned and secondary memory is allocated at this level

> Considered to be part of the operating system

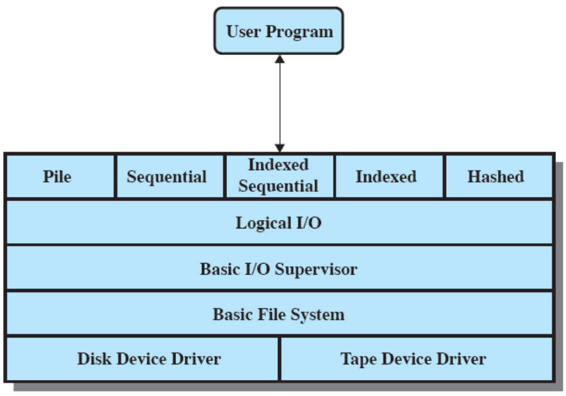
Logical I/O

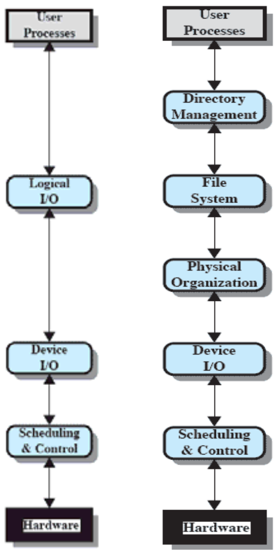
- Enables users and applications to access records as **a byte stream**
- Provides general-purpose record I/O capability
- Maintains **basic data about file**

Access Method(访问方法)

- Level of the file system closest to the user
- Provides **a standard interface** between applications and the file systems and devices that hold the data
- Different access methods reflect different **file structures** and different **ways of accessing** and processing the data

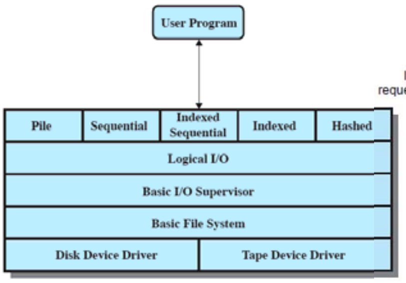
File System Software Architecture vs. I/O

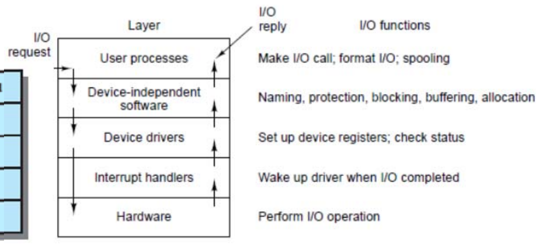




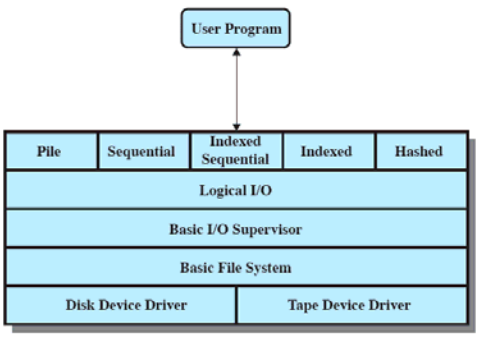
(a) Local peripheral device (c) File system

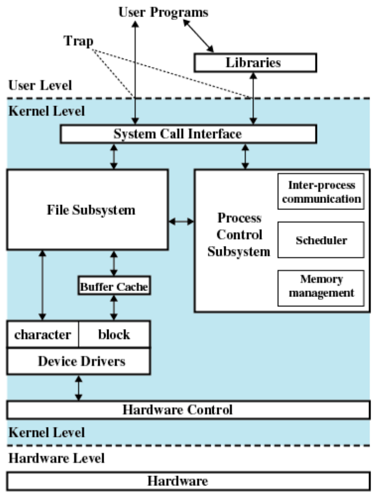
File System Software Architecture vs. I/O.





File System Software Architecture vs. I/O..





Elements of File Management

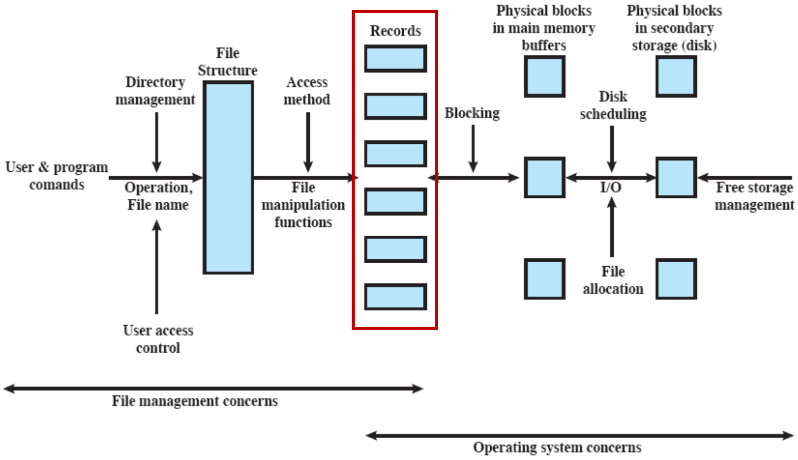


Figure 12.2 Elements of File Management

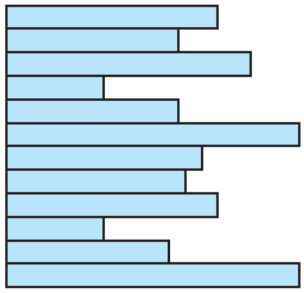
9.2 File Organization And Access

1 File Logical Organization

- File organization is the logical structuring of the records as determined by the way in which they are accessed
- In choosing a file organization, several criteria are important:
  - short access time
  - ease of update
  - economy of storage
  - simple maintenance
  - reliability

(1) The Pile

- Data are collected in **the order they arrive**
- Each record consists of **one burst of data**
- Least complicated form of file organization
- Purpose is simply to accumulate the mass of data and save it
- Record access is by **exhaustive search**

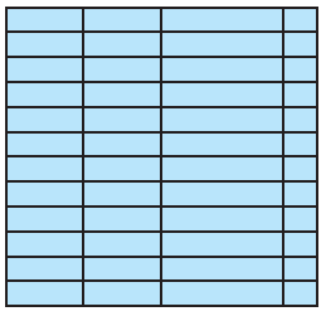


Variable-length records
Variable set of fields
Chronological order

(a) Pile File

(2) The Sequential File

- A **fixed** format is used for records
- **Key field(关键域)** uniquely identifies the record
- Most common form of file structure
- Typically used in **batch applications**
- Only organization that is easily stored on tape as well as disk

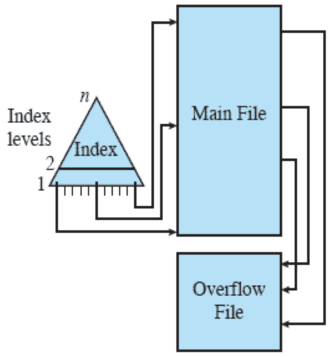


Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

(3) Indexed Sequential File

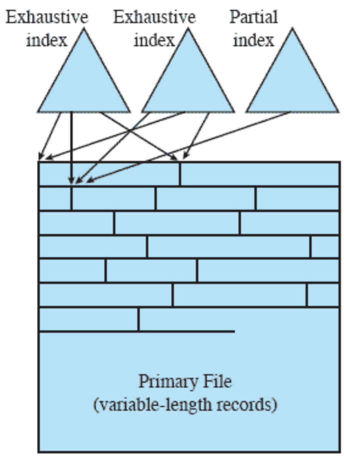
- Records are organized in **sequence** based on a key field
- Adds an **index** to the file to support random access
- Adds an **overflow file**(溢出文件) as the **log file**
- Greatly reduces the time required to access a single record
- **Multiple levels of indexing** can be used to provide greater efficiency in access



(c) Indexed Sequential File

(4) Indexed File

- Employs **multiple indexs**
  - Break the limitation: effective processing is based on a single field of file
- **Exhaustive index(完全索引)** contains one entry for every record in the main file
- **Partial index(部分索引)** contains entries to records where the field of interest exists
- Used mostly in applications where timeliness of information is critical
- Examples would be airline reservation systems and inventory control systems



(d) Indexed File

(5) Direct or Hashed File

- Access directly any block of a **known address**
- Makes use of hashing on the key value
- Often used where:
  - very rapid access is required
  - fixed-length records are used
  - records are always accessed one at a time
- Examples are:
  - directories
  - pricing tables
  - schedules
  - name lists

Grades of Performance

Table 12.1 Grades of Performance for Five Basic File Organizations [WIED87]

	Space Attributes		**Update** Record Size		**Retrieval**		
File Method	Variable	Fixed	Equal	Greater	Single record	Subset	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

Elements of File Management

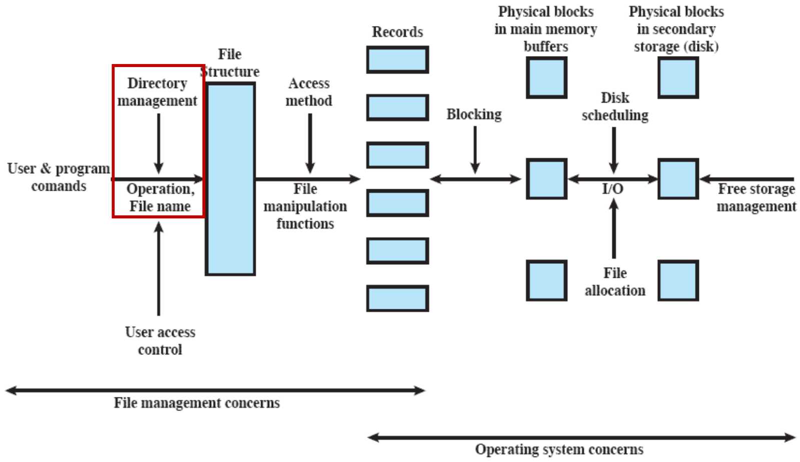


Figure 12.2 Elements of File Management

9.2 File Organization And Access

2 File Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- **Directory(目录)** itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

Direct Organization

- To understand the requirements for a file structure, it is helpful to consider the types of operations that may be performed on the directory
  - Search
  - Create files
  - Delete files
  - List directory
  - Update directory

- The simple list is not suited to supporting these operations.
- The problem is much worse in a shared system. Unique naming becomes a serious problem.

Two-Level Scheme

- There is one directory for each user and a master directory
  - **Master directory** has an entry for each user directory providing address and access control information
  - Each **user directory** is a simple list of the files of that user. Names must be unique only within the collection of files of a single user
- File system can easily enforce access restriction on directories

Tree-Structured Directory

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

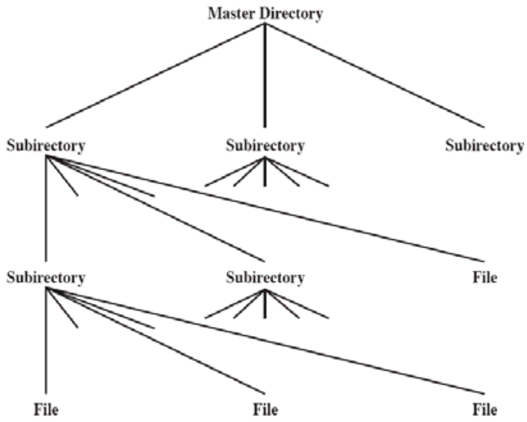
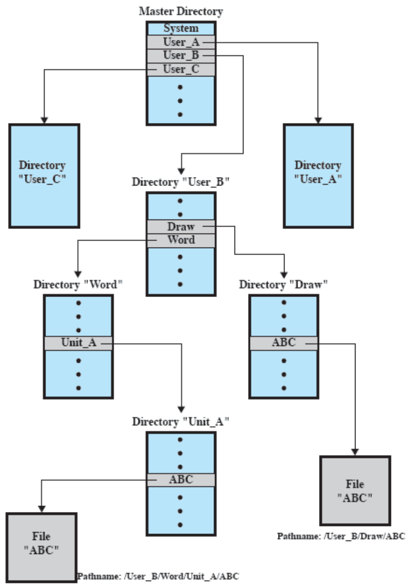


Figure 12.4 Tree-Structured Directory

"Pathname" by Tree-Structured Directory

- Files can be located by following a path from the **root**, or **master directory** down various branches. This is the **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



Elements of File Management

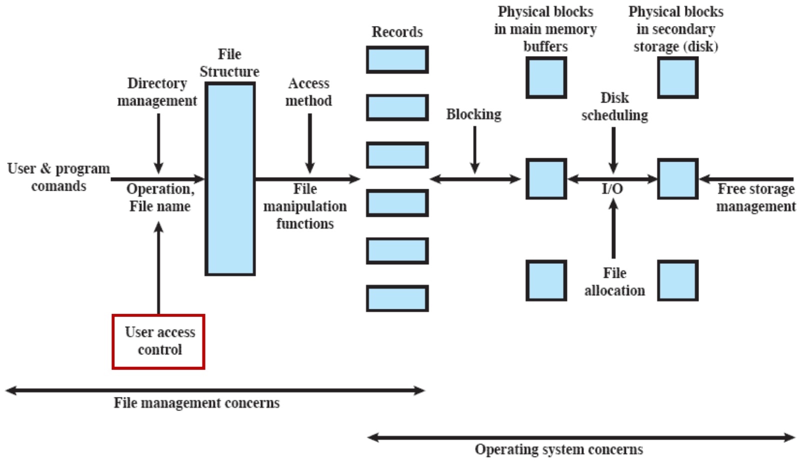


Figure 12.2 Elements of File Management

9.2 File Organization And Access

3 File Sharing

- In multiuser system, allow files to be shared among users
- Two issues
  - Access rights
  - Management of simultaneous access

Access Rights

- None
  - the user would not be allowed to read the user directory that includes the file
- Knowledge(知道)
  - the user can determine that the file exists and who its owner is and can then petition the owner for additional 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

Access Rights.

- Appending
  - the user can add data to the file but cannot modify or delete any of the file's contents
- Updating
  - the user can modify, delete, and add to the file's data
- Changing protection
  - the user can change the access rights granted to other users
- Deletion
  - the user can delete the file from the file system

- These rights can be considered to constitute a hierarchy, with each right implying those that precede it.

User Access Rights

- Owners: usually the initial creator of the file
  - has full rights previously listed
  - may grant rights to others
- Specific Users
  - individual users who are designated by user ID
- User Groups
  - a set of users who are not individually defined
- All
  - all users who have access to this system
  - these are public files

Elements of File Management

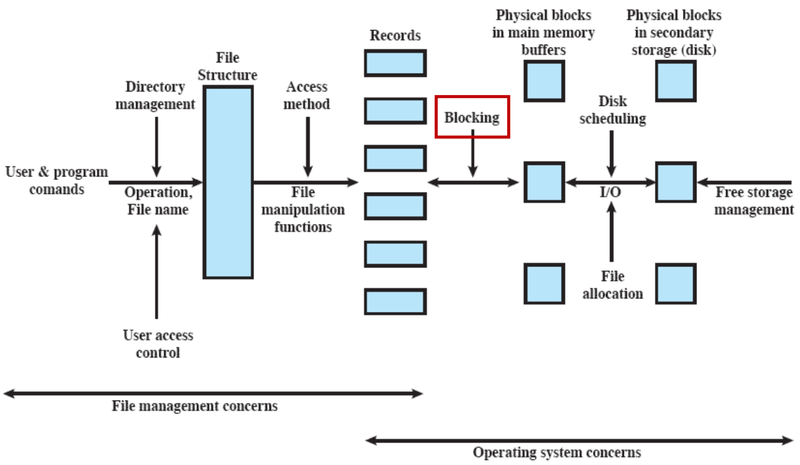


Figure 12.2 Elements of File Management

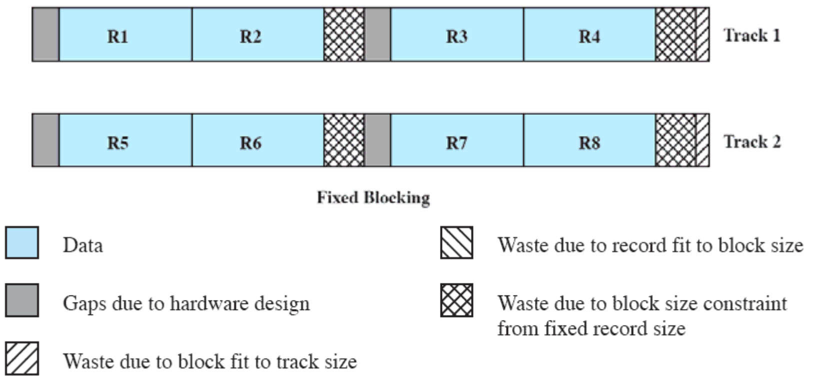
9.3 File Physical Organization

1 Record Blocking (记录组块)

- **Blocks are the unit of I/O** with secondary storage (vs. process in memory)
- for I/O to be performed records must be organized as blocks
- Given the size of a block, three methods of blocking can be used:
  - Fixed-Length Blocking
  - Variable-Length Spanned Blocking
  - Variable-Length Unspanned Blocking

Fixed Blocking

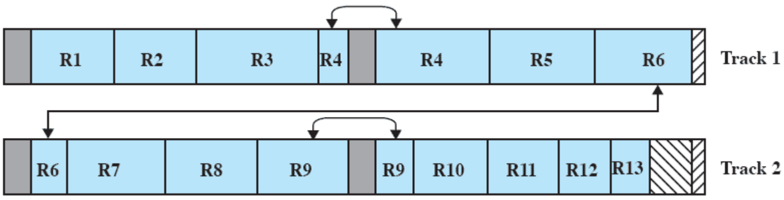
- Fixed-Length Blocking — fixed-length records are used, and an integral number of records are stored in a block
  - Internal fragmentation – unused space at the end of each block



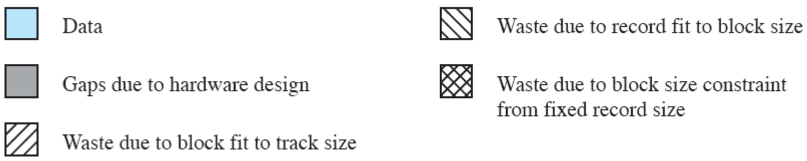
Fixed Blocking

Variable Blocking: Spanned

- Variable-Length Spanned Blocking – variable-length records are used and are packed into blocks with no unused space

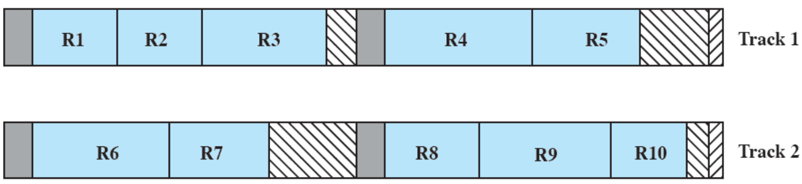


Variable Blocking: Spanned

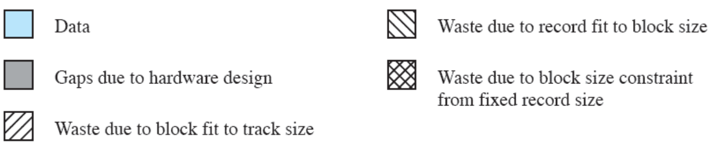


Variable Blocking: Unspanned

- Variable-Length Unspanned Blocking – variable-length records are used, but spanning is not employed



Variable Blocking: Unspanned



Elements of File Management

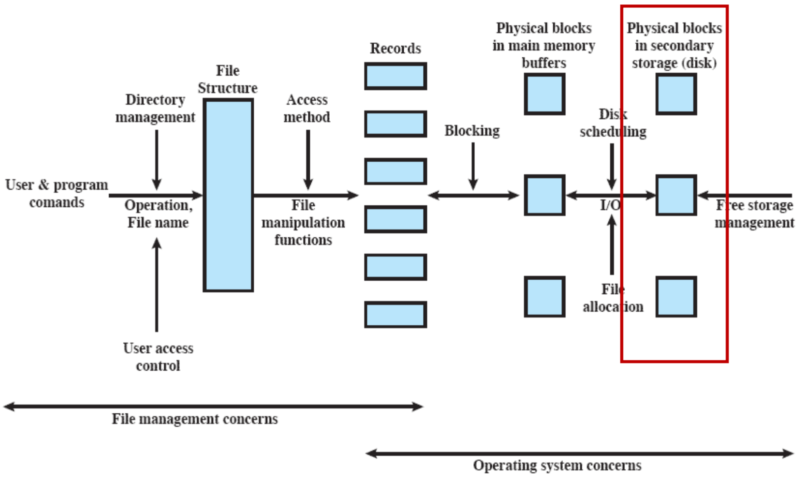


Figure 12.2 Elements of File Management

9.3 File Physical Organization

2 File Allocation (文件分配)

- The operating system or file management system is responsible for allocating blocks to files
  - Space must be allocated to files
  - Must keep track of the space available for allocation
- File allocation involves:
  - When a file created, is allocate the maximum space required?
  - What size of **portion**(contiguous set of allocated blocks) should be used?
  - What sort of data structure or table is used to keep track of the portions assigned to a file? eg. File allocation table (FAT)

Preallocation vs Dynamic Allocation

- A **preallocation policy** requires that the maximum size of a file be declared at the time of the file creation request
  - For many applications it is difficult to estimate reliably the maximum potential size of the file
  - Tends to be wasteful because users and application programmers tend to overestimate size
- **Dynamic allocation** allocates space to a file in portions as needed

Portion Size

- In choosing a portion size there is a **trade-off** between efficiency from the point of view of a single file versus overall system efficiency
- Items to be considered:
  - contiguity(邻近) of space increases performance, especially for Retrieve_Next operations
  - having a large number of small portions increases the size of tables needed to manage the allocation information
  - having fixed-size portions simplifies the reallocation of space
  - having variable-size or small fixed-size portions minimizes waste of unused storage due to overallocation (vs. page)

Two Alternatives

- Variable, large **contiguous portions**
  - provides better performance
  - the variable size avoids waste
  - the file allocation tables are small
- **Blocks**
  - small fixed portions provide greater flexibility
  - they may require large tables or complex structures for their allocation
  - contiguity has been abandoned as a primary goal
  - blocks are allocated as needed

9.3 File Physical Organization

3 File Physical Organization

- The file physical organization refers to how the file is stored on a peripheral devices. It is related to the performance of the storage.

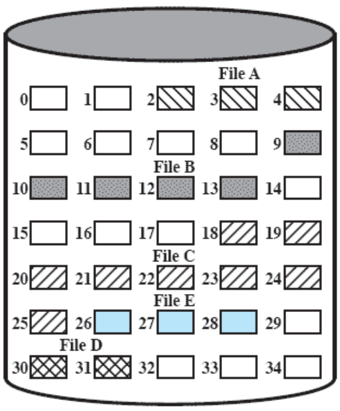
	Contiguous	**Chained**	**Indexed**	
Preallocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table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

Table 12.3 File Allocation Methods

(1) Contiguous Allocation—Contiguous File

- Single **contiguous set** of blocks is allocated to a file at the time of creation
  - Preallocation strategy using variable-size portions
  - Is the best from the point of view of the individual sequential file
- The file allocation table needs just **a single entry** for each file
  - Starting block and length of the file
- External fragmentation will occur
  - Need to perform compaction

Contiguous Allocation

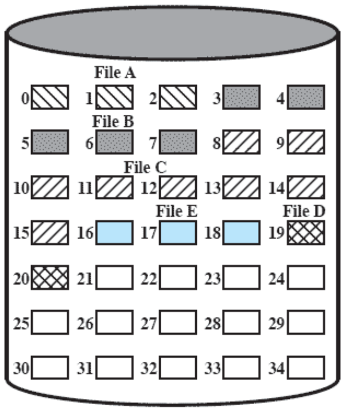


File Allocation Table

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

After Compaction (紧缩)



File Allocation Table

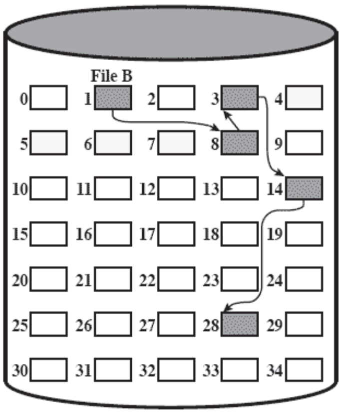
File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

Figure 12.8 Contiguous File Allocation (After Compaction)

(2) Chained Allocation—Chained File

- Allocation on basis of **individual block**
- Each block contains **a pointer** to the next block in the chain
- The file allocation table needs just **a single entry** for each file
  - Starting block and length of file
- No external fragmentation
- Best for sequential files
- No accommodation of the principle of locality

Chained Allocation

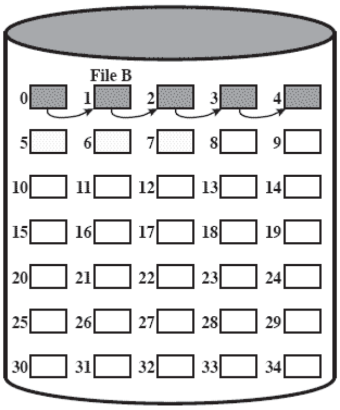


File Allocation Table

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

Figure 12.9 Chained Allocation

Chained Allocation After Consolidation(合并)



File Allocation Table

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

Figure 12.10 Chained Allocation (After Consolidation)

(3) Indexed Allocation—Indexed File

- File allocation table contains a separate **one-level index** for each file
- The index has **one entry** for each portion allocated to the file
  - Indexed allocation with **blocks**: The file allocation table contains block number for the index
  - Indexed allocation with **variable-length portions**

With Blocks

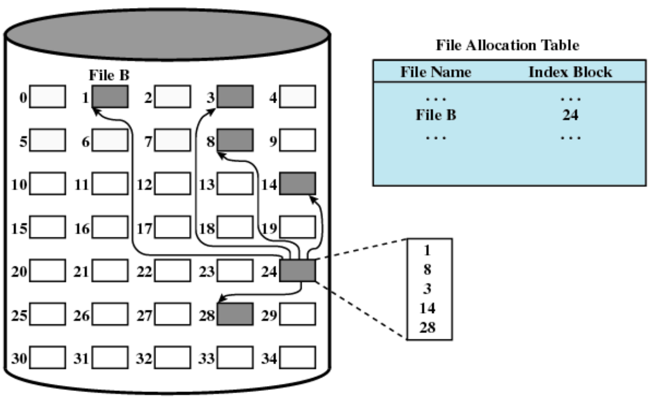


Figure 12.11 Indexed Allocation with Block Portions

With Variable-Length Portions

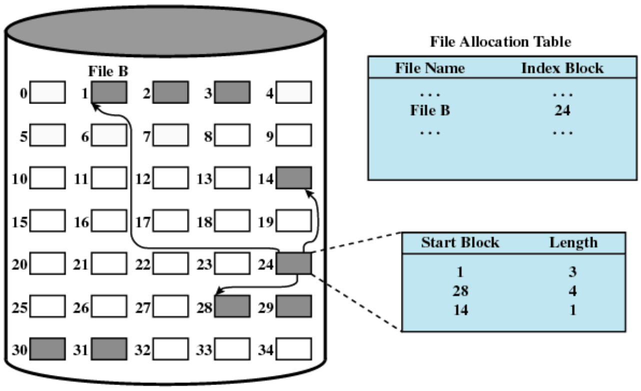


Figure 12.12 Indexed Allocation with Variable-Length Portions

Elements of File Management

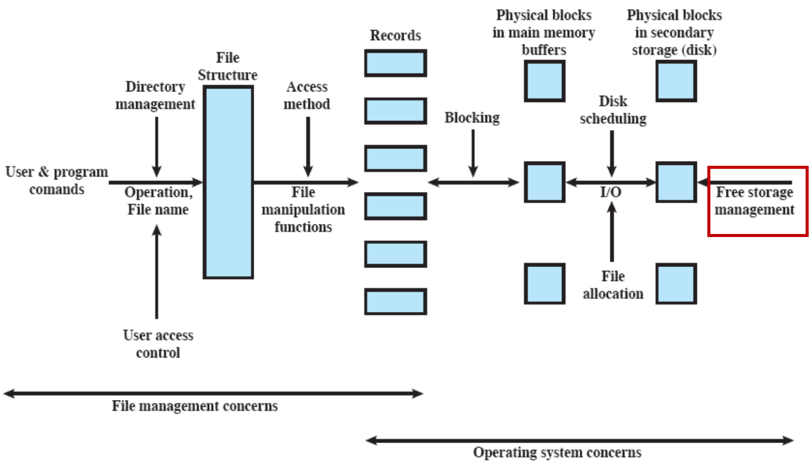


Figure 12.2 Elements of File Management

9.3 File Physical Organization

4 Free Space Management

- DAT (Disk available table)
  - Bit table
  - Chained Free Portions
  - Indexing
  - Free Block List

Bit Tables (位表)

- A vector containing **one bit for each block** on the disk
- Each entry of a 0 corresponds to a free block, and each 1 corresponds to a block in use
- Keep in memory
- Advantages:
  - works well with any file allocation method
  - it is as small as possible

Chained Free Portions (链接空闲区)

- The free portions may be chained together by using **a pointer** and **length value** in each free portion
- Advantages:
  - Negligible space overhead because there is no need for a disk allocation table
  - Suited to all file allocation methods
- Disadvantages:
  - leads to fragmentation
  - every time you allocate a block you need to read the block first to recover the pointer to the new first free block before writing data to that block

Indexing (索引)

- Treats free space as a file and uses an index table as it would for file allocation
- For efficiency, the index should be **on the basis of variable-size portions** rather than blocks
- This approach provides efficient support for all of the file allocation methods

Free Block List (空闲块表)

- Each block is assigned a number sequentially
  - the list of the numbers of all free blocks is maintained in a reserved portion of the disk
- Depending on the size of the disk, either 24 or 32 bits will be needed to store a single block number
  - the size of the free block list is 24 or 32 times the size of the corresponding bit table and must be stored on disk
- There are two effective techniques for storing a small part of the free block list in main memory:
  - the list can be treated as a push-down stack with the first few thousand elements of the stack kept in main memory
  - the list can be treated as a FIFO queue, with a few thousand entries from both the head and the tail of the queue in main memory

9.4 UNIX File Management

Types of files

- Regular, or ordinary
- Directory
- Special
- Named pipes
- Links
- Symbolic links

Inodes

- Index node (索引节点)
- Control structure that contains key information for a particular file

Layout of Unix File on Disk

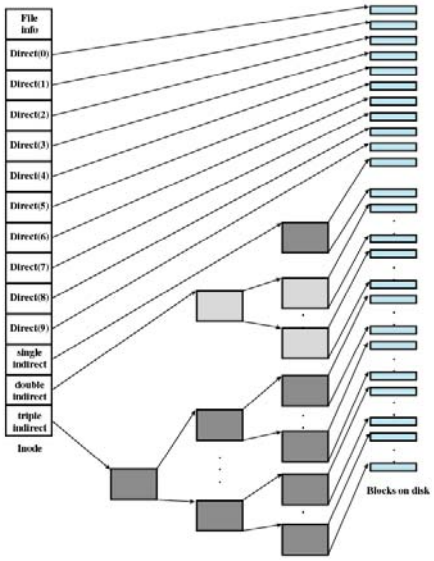
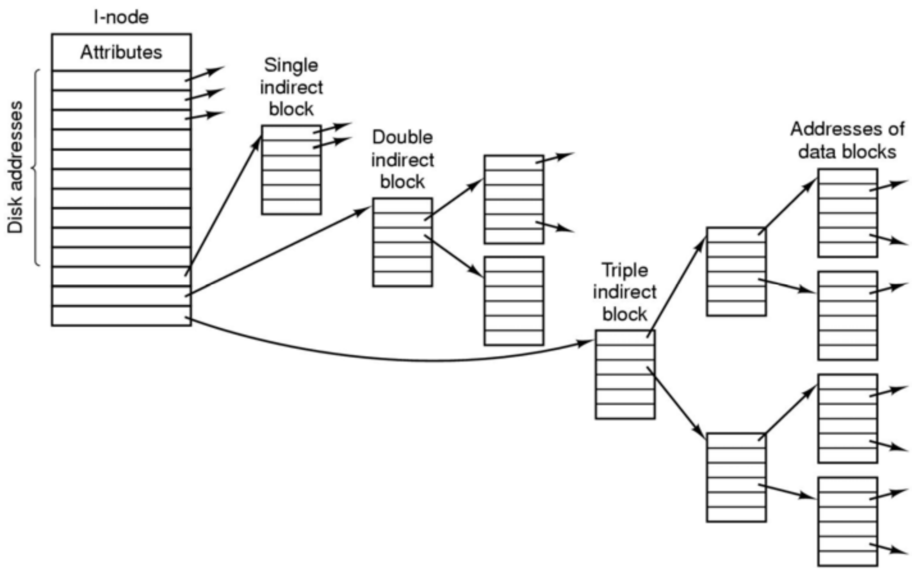


Figure 12.13 Layout of a UNIX File on Disk

Layout of Unix File on Disk.



Unix Directory and I-node

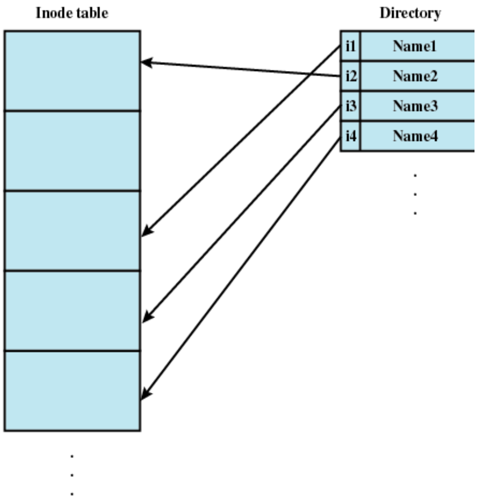


Figure 12.14 UNIX Directories and Inodes

Unix Pathname Parsing

- /usr/ast/mbox

Root dir

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

i-node 6

Attirb
132

Block 132

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

i-node 26

Attrib
406

Block 406

26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

Free Space Management

- Bitmap
- Chained (Linked List)

42	230	86
196	162	237
210	612	897
97	342	422
41	214	140
63	160	223
21	664	223
48	216	160
262	320	126
…	…	…
310	180	142
516	462	141

Terminology

- field; record
- file; database
- file system(file management system)
- file organization
  - pile
  - sequential file
  - indexed sequential file
  - indexed file
  - hashed file
- directory
  - pathname
  - root(master directory); working directory(current directory)

Terminology

- portion
- file allocation methods
  - contiguous allocation
  - chained allocation
  - indexed allocation
- free space management
  - bit table
  - chained free portion
  - indexing
  - free block list
- inode