Chapter 12
File Management
Contents

- Overview
- File organization and Access
- Record blocking
- Secondary storage management
- Unix file management
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
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
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
  - 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 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

Exhaustive index

Partial index

Primary File (variable-length records)
File Organization

- 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

- Direct, or Hashed, file

Hash Function

Key \rightarrow f

Primary File

Overflow File
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

Track 1

R1 | R2 | R3 | R4

Track 2

R5 | R6 | R7 | R8

Fixed Blocking

- Data
- Gaps due to hardware design
- Waste due to block fit to track size
- Waste due to record fit to block size
- Waste due to block size constraint from fixed record size
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

- Data
- Gaps due to hardware design
- Waste due to record fit to block size
- Waste due to block size constraint from fixed record size
- Waste due to block fit to track size
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

- **Track 1**
  - R1
  - R2
  - R3
  - R4
  - R5

- **Track 2**
  - R6
  - R7
  - R8
  - R9
  - R10

Legend:
- **Data**
- **Gaps due to hardware design**
- **Waste due to block fit to block size**
- **Waste due to block size constraint from fixed record size**
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)
    - 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
  - Single contiguous set of blocks is allocated to a file at the time of creation
  - 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
Figure 12.7 Contiguous file allocation
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
Figure 12.10  Chained allocation (after consolidation)
Methods of File Allocation

• Indexed allocation: Unix file system
  • File allocation table contains block number for the index
    – Index block has one entry for each portion allocated to the file
Figure 12.11  Indexed allocation with block portions
Figure 12.12 Indexed allocation with variable-length portions
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
    – 0011100001111100001111111111011000

● Easy to find free blocks
● It is as small as possible
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
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

Disk Drive

Partition

Partition

Partition

File System

i-list

Directory Blocks and Data Blocks

Boot Block(s)

Super Block

i-list

i-node

i-node

\ldots

i-node
UNIX File System: I-node

- File owner, group 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
Table 12.5 Capacity of a UNIX File

<table>
<thead>
<tr>
<th>Level</th>
<th>Number of Blocks</th>
<th>Number of Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>10</td>
<td>10K</td>
</tr>
<tr>
<td>Single Indirect</td>
<td>256</td>
<td>256K</td>
</tr>
<tr>
<td>Double Indirect</td>
<td>$256 \times 256 = 65K$</td>
<td>65M</td>
</tr>
<tr>
<td>Triple Indirect</td>
<td>$256 \times 65K = 16M$</td>
<td>16G</td>
</tr>
</tbody>
</table>