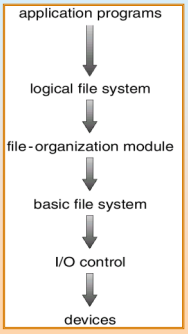
Chapter 12: File System Implementation

- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery
- Log-Structured File Systems

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks).
- File system organized into layers.
- *File control block* – storage structure consisting of information about a file.

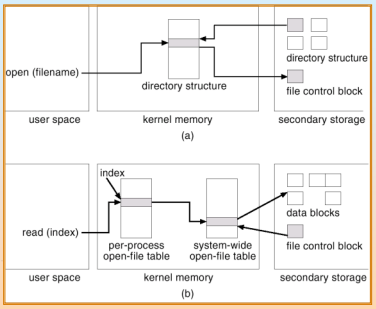
Layered File System



A Typical File Control Block

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks

In-Memory File System Structures



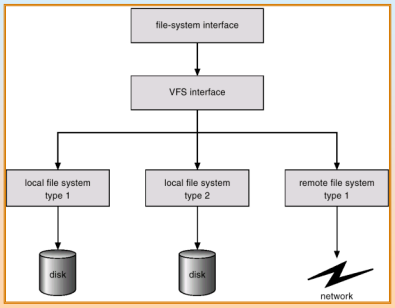
- Figure 12-3(a) refers to opening a file.
- Figure 12-3(b) refers to reading a file.

Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API* is to the VFS interface, rather than any specific type of file system.

API - Application Program Interface

Schematic View of Virtual File System



Directory Implementation

- Linear list of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- Hash Table – linear list with hash data structure.
  - decreases directory search time
  - *collisions* – situations where two file names hash to the same location [-> linked list]

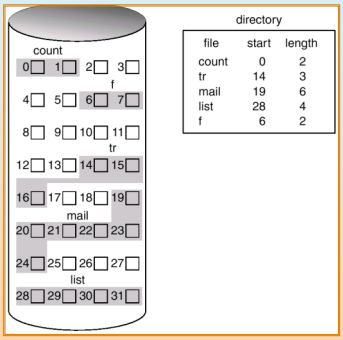
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation
- Linked allocation
- Indexed allocation

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required.
- Random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

Contiguous Allocation of Disk Space



Extent-Based Systems

- Many newer file systems (I.e. Veritas File System) use a modified contiguous allocation scheme.
- Extent-based file systems allocate disk blocks in **extents**.
- An **extent** is a contiguous block of disks. Extents are allocated for file allocation. A file consists of one or more extents.

Linked Allocation

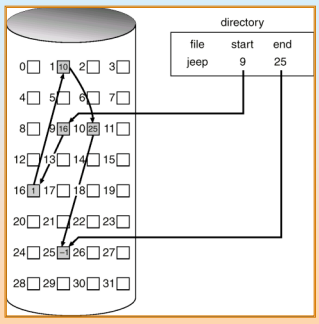
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

block = | pointer |

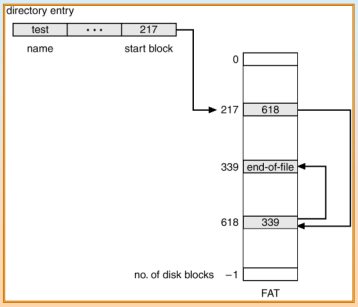
Linked Allocation (Cont.)

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access

Linked Allocation

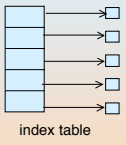


File-Allocation Table

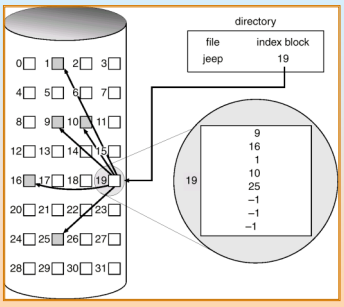


Indexed Allocation

- Brings all pointers together into the *index block*.
- Logical view.



Example of Indexed Allocation



Indexed Allocation – Mapping (Cont.)

- Two-level index (maximum file size is 512^3)

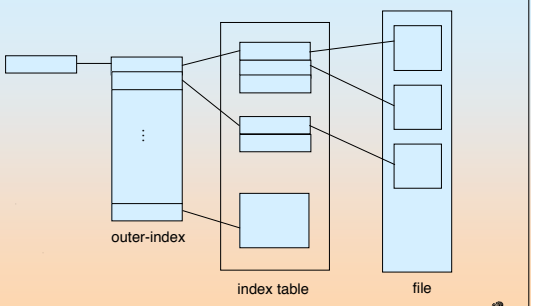
$$LA / (512 \times 512) \begin{cases} Q_1 \\ R_1 \end{cases}$$

Q_1 = displacement into outer-index
R_1 is used as follows:

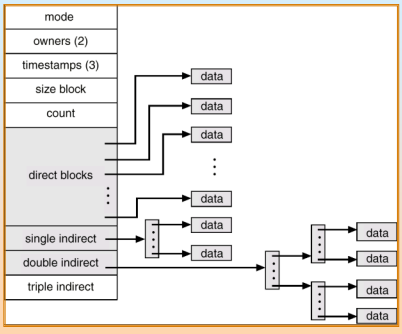
$$R_1 / 512 \begin{cases} Q_2 \\ R_2 \end{cases}$$

Q_2 = displacement into block of index table
R_2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)



Combined Scheme: UNIX (4K bytes per block)



Free-Space Management

- Bit vector (*n*-bit word, represents *n* blocks)

If bit[*i*] = { 0 ⇒ block[*i*] free
1 ⇒ block[*i*] occupied

Block number calculation

(number of bits per word) *(number of words) + offset of bit

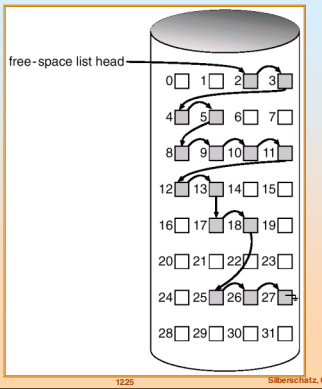
Free-Space Management (Cont.)

- Bit map requires extra space. Example:
  block size = 2^{12} bytes
  disk size = 2^{30} bytes (1 gigabyte)
  $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
- Easy to get contiguous files
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping [list of 1st n free blocks in 1st block]
- Counting [indicate 1st free block and number of consecutive free blocks]

Directory Implementation

- Linear list of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- Hash Table – linear list with hash data structure.
  - decreases directory search time
  - *collisions* – situations where two file names hash to the same location [-> linked list]
  - fixed size

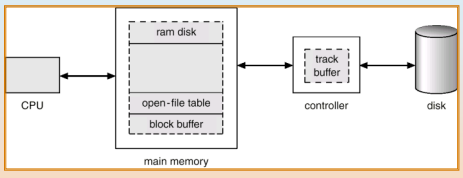
Linked Free Space List on Disk



Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks
  - free-behind and read-ahead – techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk [only benefits processes using this file, reduces memory for general use].

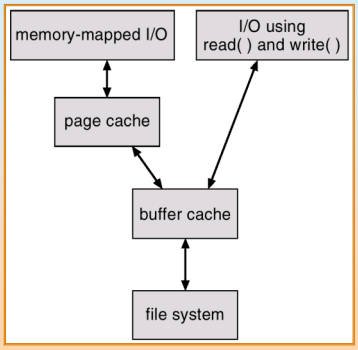
Various Disk-Caching Locations



Page Cache

- A **page cache** caches pages rather than disk blocks using virtual memory techniques.
- Memory-mapped I/O uses a page cache.
- Routine I/O through the file system uses the buffer (disk) cache.
- This leads to the following figure.

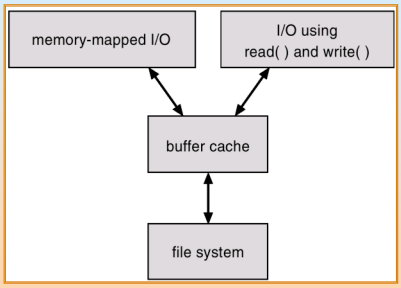
I/O Without a Unified Buffer Cache



Unified Buffer Cache

- A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O.

I/O Using a Unified Buffer Cache



Recovery

- Consistency checking – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.
- Use system programs to *back up* data from disk to another storage device (floppy disk, magnetic tape).
- Recover lost file or disk by *restoring* data from backup.

Log Structured File Systems

- **Log structured** (or journaling) file systems record each update to the file system as a **transaction**. [Linux added Journaling last year]
- All transactions are written to a **log**. A transaction is considered **committed** once it is written to the log. However, the file system may not yet be updated.
- The transactions in the log are asynchronously written to the file system. When the file system is modified, the transaction is removed from the log.
- If the file system crashes, all remaining transactions in the log must still be performed.