



Operating Systems

File Systems

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing - Protection

File Concept

- Contiguous logical address space
- Types:
 - Data
 - numeric
 - character
 - binary
 - Program

File Structure

- None - sequence of words, bytes
- Simple record structure
 - Lines
 - Fixed length
 - Variable length
- Complex Structures
 - Formatted document
 - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters.
- Who decides:
 - Operating system
 - Program

File Attributes

- Name**
- Type** – needed for systems that support different types.
- Location** – point of file location on the device
- Size** – current file size.
- Protection** – controls who can do reading, writing, executing.
- Time, date, and user identification** security, and usage monitoring.
- Information about files are kept in the directory structure, which is maintained on the disk.

File Operations

- Create
- Write
- Read
- Reposition within file – file seek
- Delete
- Truncate
- Open(F_i) – search the directory structure on disk for entry F_i , and move the content of entry to memory.
- Close (F_i) – move the content of entry F_i in memory to directory structure on disk.

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rft, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Access Methods

- Sequential Access**
 - read next*
 - write next*
 - reset*
 - no read after last write (rewrite)*
 - Direct Access**
 - read n*
 - write n*
 - position to n*
 - read next*
 - write next*
 - rewrite n*
- n* = relative block number

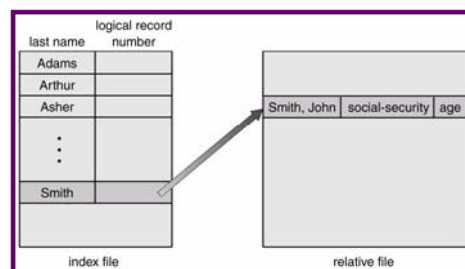
Sequential-access File



Simulation of Sequential Access on a Direct-access File

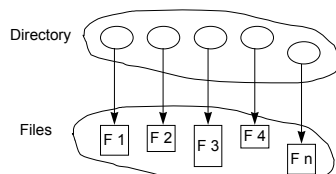
sequential access	implementation for direct access
<i>reset</i>	<i>cp = 0;</i>
<i>read next</i>	<i>read cp;</i> <i>cp = cp+1;</i>
<i>write next</i>	<i>write cp;</i> <i>cp = cp+1;</i>

Example of Index and Relative Files



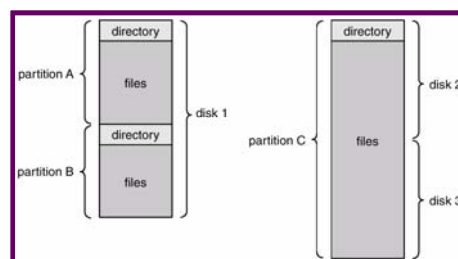
Directory Structure

- A collection of nodes containing information about all files.



Both the directory structure and the files reside on disk. Backups of these two structures are kept on tapes.

A Typical File-system Organization



Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information (discuss later)

Operations Performed on Directory

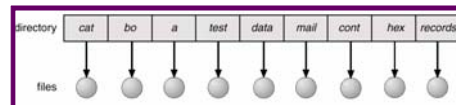
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Organize the Directory (Logically) to Obtain ...

- **Efficiency**
- **Naming** – convenient to users.
 - Two users can have same name for different files.
 - The same file can have several different names.
- **Grouping** – logical grouping of files by properties, (e.g., all Java programs, all games, ...)

Single-Level Directory

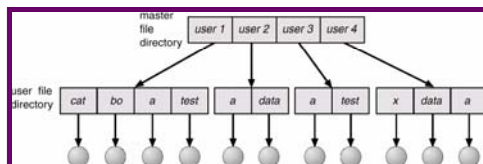
- A single directory for all users.



- naming problem
- grouping problem

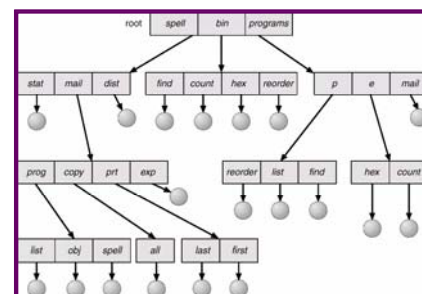
Two-Level Directory

- Separate directory for each user.



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
 - `cd /spell/mail/prog`
 - `type list`

Tree-Structured Directories (Cont.)

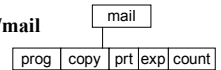
- **Absolute** or **relative** path name
- Creating a new file is done in current directory.
- Delete a file


```
rm <file-name>
```
- Creating a new subdirectory is done in current directory.

`mkdir <dir-name>`

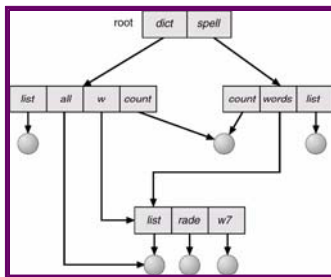
Example: if in current directory `/mail`

`mkdir count`



- Deleting "mail"
 - ⇒ deleting the entire subtree rooted by "mail".

Acyclic-Graph Directories

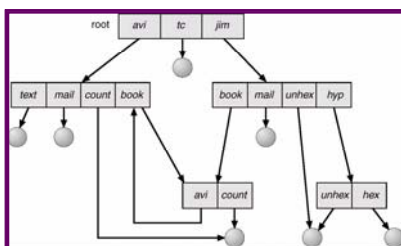


- Have shared subdirectories and files.

Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If `dict` deletes `list` ⇒ dangling pointer.
 - Solutions:
 - Backpointers, so we can delete all pointers. Variable size records a problem.
 - Backpointers using a daisy chain organization.
 - Entry-hold-count solution.

General Graph Directory



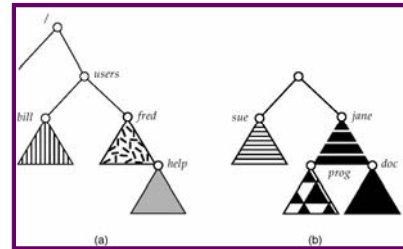
General Graph Directory (Cont.)

- How do we guarantee no cycles?
 - Allow only links to file not subdirectories.
 - Garbage collection.
 - Every time a new link is added use a cycle detection algorithm to determine whether it is OK.

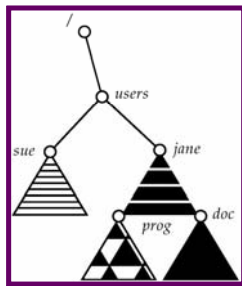
File System Mounting

- A file system must be **mounted** before it can be accessed.
- A unmounted file system (I.e. Fig. 11-11(b)) is mounted at a **mount point**.

(a) Existing. (b) Unmounted Partition



Mount Point



File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a *protection* scheme.
- On distributed systems, files may be shared across a network.
- Network File System (NFS) is a common distributed file-sharing method.

Protection

- File owner/creator should be able to control:
 - what can be done
 - by whom
- Types of access
 - Read
 - Write
 - Execute
 - Append
 - Delete
 - List

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) owner access	7	⇒	1 1 1
b) group access	6	⇒	1 1 0
			RWX
c) public access	1	⇒	0 0 1
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.

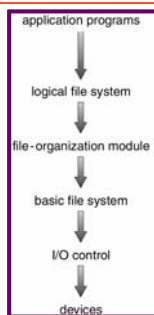
File System Implementation

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

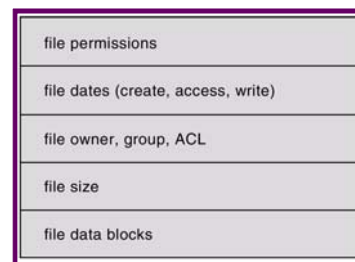
File-System Structure

- 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.

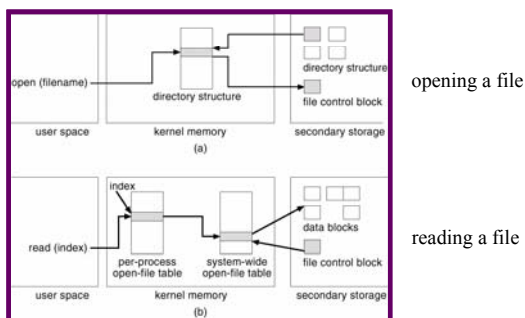
Layered File System



A Typical File Control Block



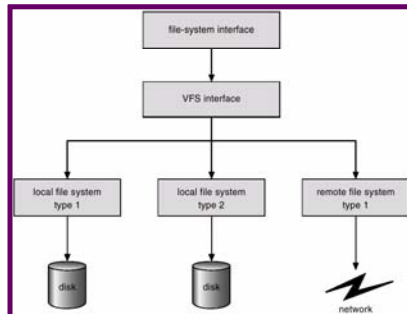
In-Memory File System Structures



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.

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
 - fixed size

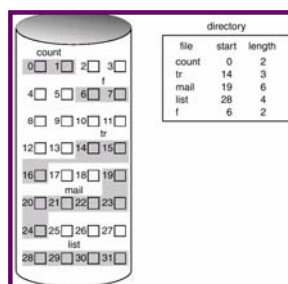
Allocation Methods

- 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

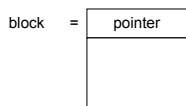


Extent-Based Systems

- 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.



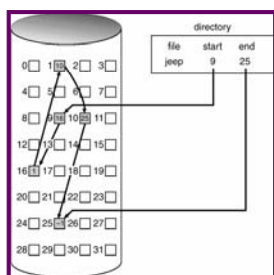
Linked Allocation (Cont.)

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

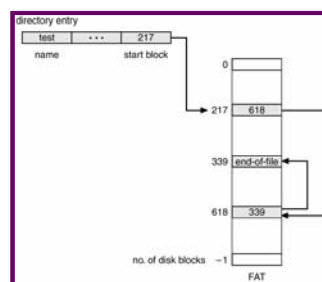


- Block to be accessed is the Qth block in the linked chain of blocks representing the file.
- Displacement into block = $R + 1$
- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.

Linked Allocation

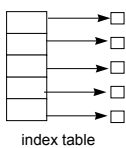


File-Allocation Table

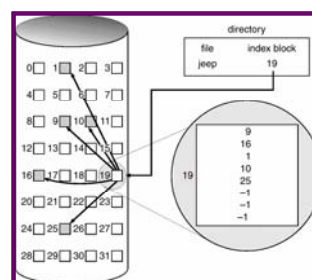


Indexed Allocation

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



Example of Indexed Allocation



Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table.



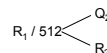
Q = displacement into index table
 R = displacement into block

Indexed Allocation – Mapping (Cont.)

- Mapping from logical to physical in a file of unbounded length (block size of 512 words).
- Linked scheme – Link blocks of index table (no limit on size).

$$= \text{block of index table} \begin{cases} Q_1 \\ R_1 \end{cases}$$

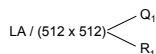
Q_1
 R_1 is used as follows:



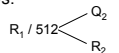
Q_2 = displacement into block of index table
 R_2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)

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

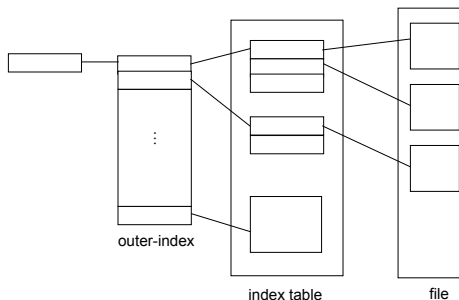


Q_1 = displacement into outer-index
 R_1 is used as follows:

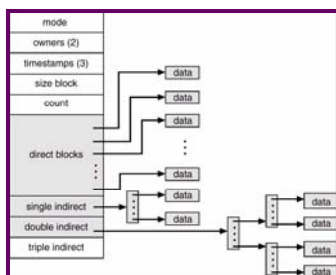


Q_2 = displacement into block of index table
 R_2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)



Combined Scheme: UNIX (4K bytes per block)



Free-Space Management

- Bit vector (n blocks)



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Block number calculation

$$\begin{aligned} & (\text{number of bits per word}) * \\ & (\text{number of 0-value words}) + \\ & \text{offset of first 1 bit} \end{aligned}$$

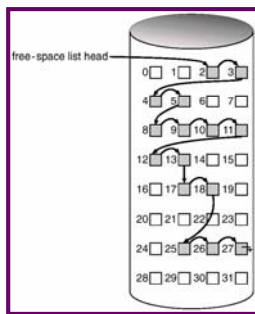
Free-Space Management (Cont.)

- 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
- Counting

Free-Space Management (Cont.)

- Need to protect:
 - Pointer to free list
 - Bit map
 - Must be kept on disk
 - Copy in memory and disk may differ.
 - Cannot allow for block[i] to have a situation where bit[i] = 1 in memory and bit[i] = 0 on disk.
 - Solution:
 - Set bit[i] = 1 in disk.
 - Allocate block[i]
 - Set bit[i] = 1 in memory

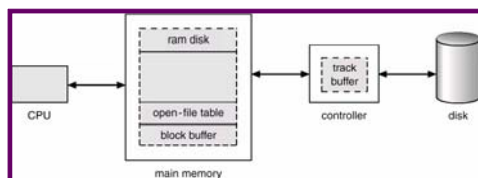
Linked Free Space List on Disk



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.

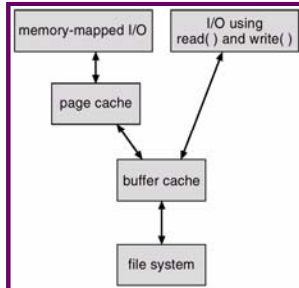
Various Disk-Caching Locations



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.

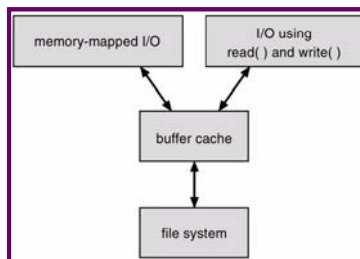
I/O Without a Unified Buffer Cache



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.