File - System Interface

1 File Concept

- Contiguous logical address space
- Types
  - Data
    - Numeric
    - Character
    - Binary
  - Program
- Contents defined by file's creator
  - Many types like text, source, executable

2 File attributes

- **Name** - Only information kept in human-readable form
- **Identifier** - Unique tag (number) identifies file within file system
- **Type** - Needed for systems that support different types
- **Location** - Pointer to the file location on device
- **Size** - Current file size
- **Protection** - Controls who can do reading, writing, executing
- **Time, date and user id** - Data for protection, security and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

3 File operations

- File is an abstract data type
- Create
- Write - At write pointer location
- Read - Ar read pointer location
- Reposition within file - seek
- Delete
- Truncate
- $Open(F_i)$ - search the directory structure on dis 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

4 Open files

- Several pieces of data are needed to manage open files:
  - Open-file table - tracks open files
  - File pointer: pointer to last read/write location, per process that has the file open
  - File-open count: counter of number of times a file is open - to allow removal of data from open-file table when last process closes it
  - Disk location of the file: cache of data access information
  - Access rights: per process access mode information

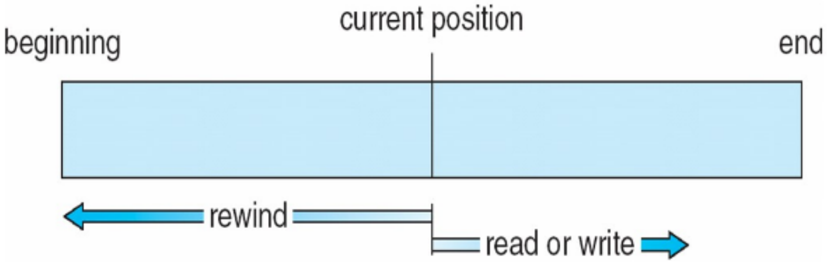
5 Open file locking

- Provided by some operating systems and file systems
  - Similar to reader-writer locks
  - Shared lock similar to reader lock - several process can acquire concurrently
  - Exclusive lock similar to writer lock
- Mediates access to a file
- Mandatory or advisory
  - Mandatory - Access is denied depending on locks held and requested
  - Advisory - Process can find status of locks and decide what to do

6 File structure

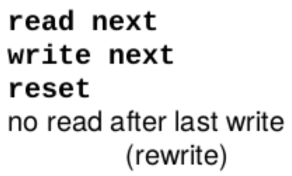
- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides
  - Operating system
  -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

7 Sequential access file



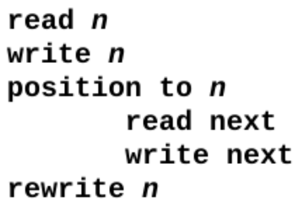
8 Access methods

- Sequential access



```
read next
write next
reset
no read after last write
        (rewrite)
```

- Direct access - file is fixed length logical records



```
read n
write n
position to n
       read next
       write next
rewrite n
```

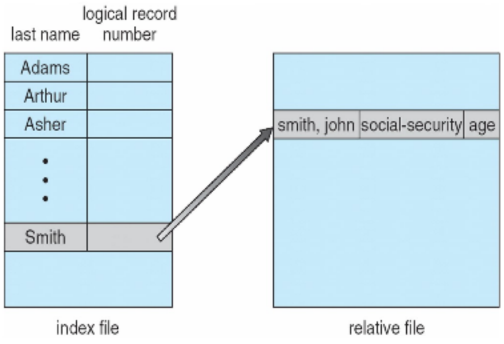
- Relative block numbers allow the OS to decide where the block should be placed

9 Simulation of Sequential access on a direct-access file

Sequential Acess	Implementation for direct access
Reset	cp=0;
Read Next	read cp; cp=cp+1;
Write next	write cp; cp=cp+1;

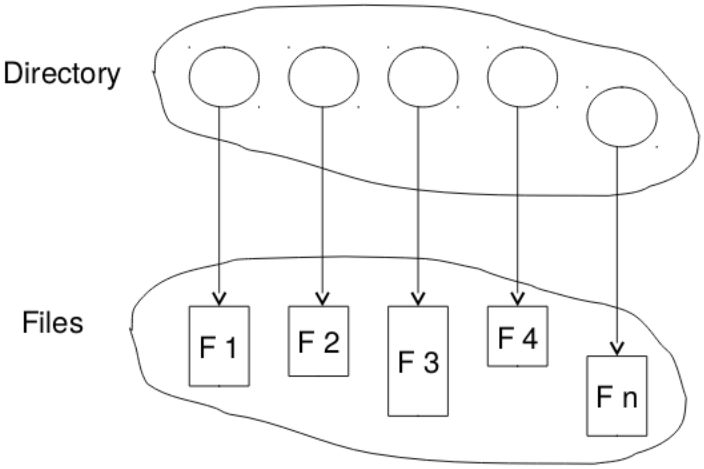
10 Other access methods

- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)



11 Directory Structure

- A collection of nodes containing information about all files

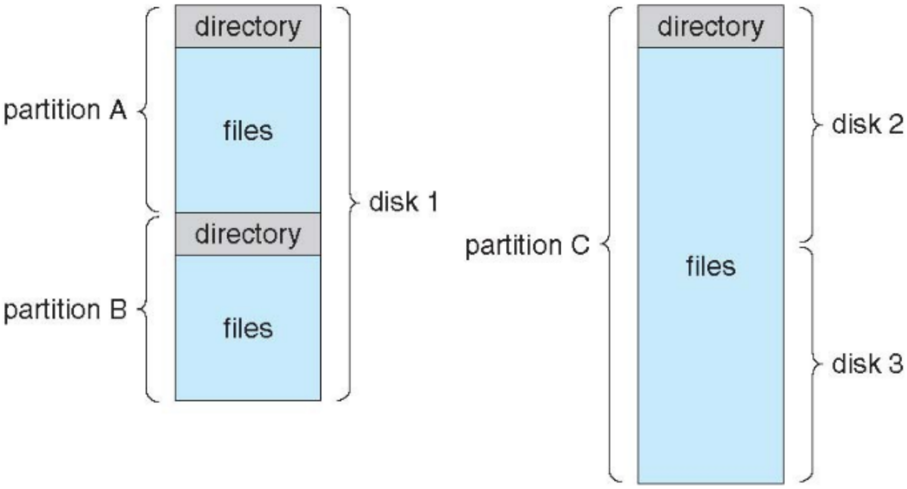


Both the directory structure and the files reside on disk

12 Disk structure

- Disk can be subdivided into partitions
- Disks or partitions can be RAID protected against failure
- Disk or partition can be used raw - without a file system, or formatted with a file system
- Partitions also known as minidisks, slices
- Entity containing file system known as volume
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents
- A well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

13 A Typical File-system Organization



14 Operations Performed on Directory

- Search for a file
- Create a file

- Delete a file
- List a directory
- Rename a file
- Traverse the file system

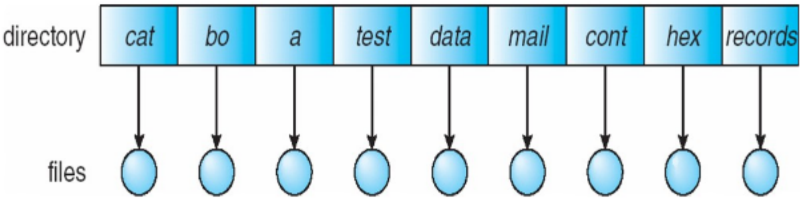
15 Directory Organisation

The directory is organized logically to obtain

- Efficiency - Locating a file quickly
- Naming - convenient to users
  - Two users can have the same name for different files
  - The same file can have several different names
- Grouping - logical grouping of files by properties

16 Single-Level Directory

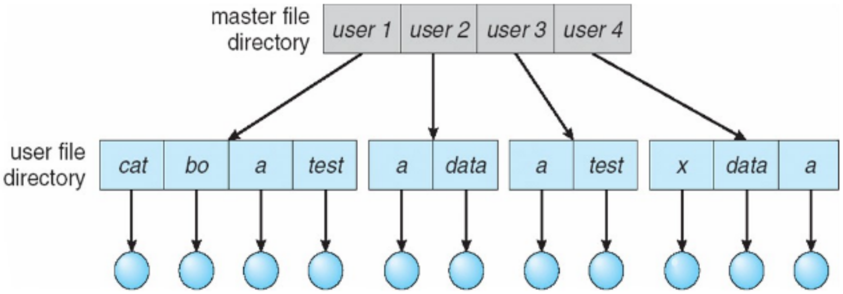
- A single directory for all users



- Naming problem
- Grouping problem

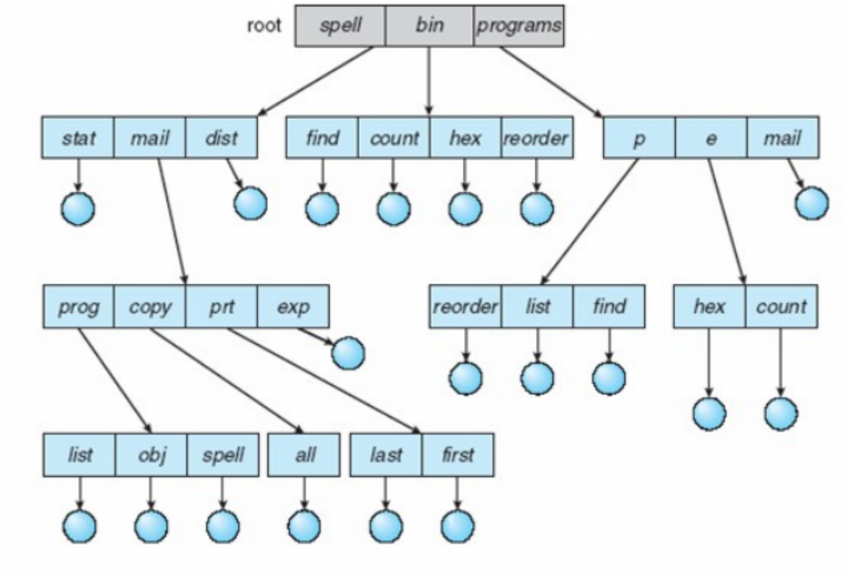
17 Two level directory

- Separate directory for each user



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

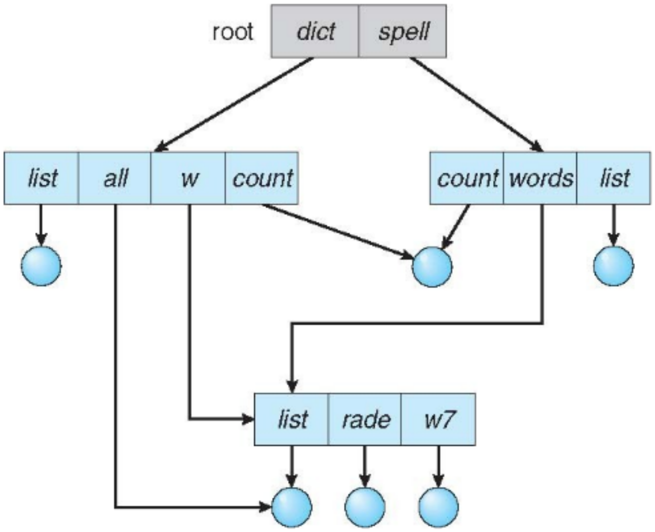
18 Tree-Structured Directories



- Efficient searching
- Grouping capability
- Absolute or relative path name
- Creating a new file is done in current directory

19 Acyclic-Graph Directories

- Have shared subdirectories and files

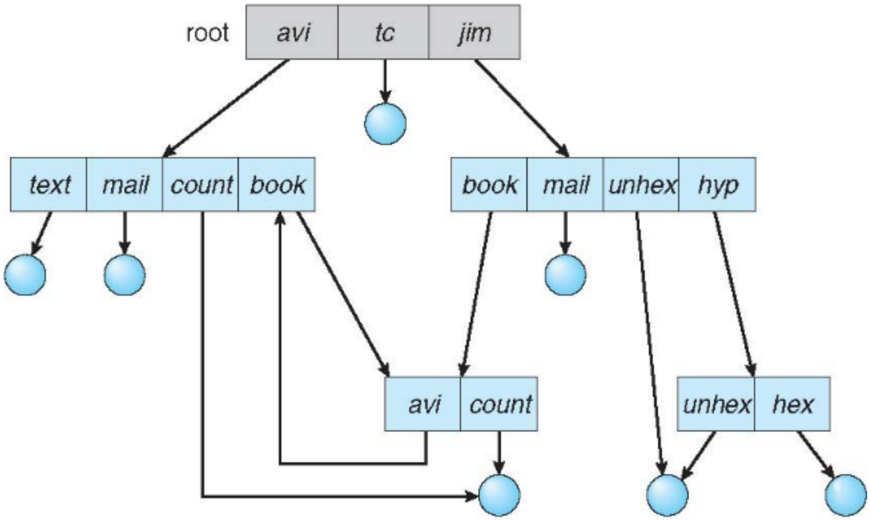


- Two different names (aliasing)
- If dict deletes list ⇒ dangling pointer
  Solutions
  - Backpointers, so we can delete all pointers. Variable size records a problem
  - Backpointers using a daisy chain organization

  - Entry-hold-count solution

- New directory entry type
  - Link - another name (pointer) to an existing file
  - Resolve the link - follow pointer to locate the file

20 General Graph Directory



- 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

21 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 Systems (NFS) is a common distributed file-sharing method
- If multi user system
  - User IDs identify users, allowing permissions and protections to be per-user
    Group IDs allow users to be in groups, permitting group access rights
  - Owner of a file/directory
  - Group of a file/directory

21.1 Failure Modes

- All file systems have failure modes. For example, corruption of directory structures or other non-user data, called metadata
- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve state information about status of each remote request
- Stateless protocols such as NFS v3 include all information in each request, allowing easy recovery but less security

21.2 Consistency Schematics

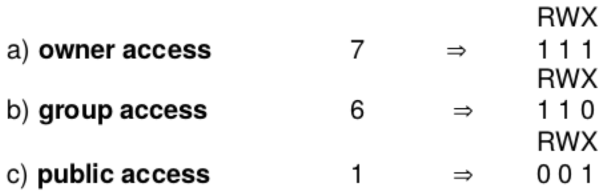
- Specify how multiple users are to access a shared file simultaneously
  - Tend to be less complex due to disk I/O and network latency
  - Andrew File System (AFS) implemented complex remote file sharing semantics
  - Unix file system (UFS) implements:
    * Writes to an open file visible immediately to other users of the same open file
    * Sharing file pointer to allow multiple users to read and write concurrently
  - AFS has session semantics - Writes only visible to sessions starting after the file is closed

22 Protection

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

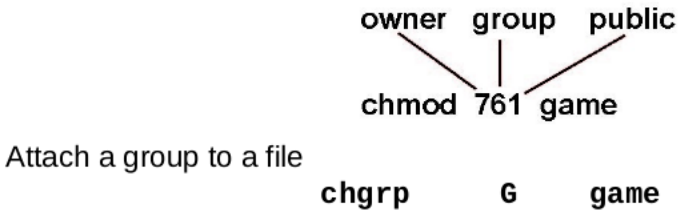
22.1 Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users on Unix/ Linux



			RWX
a) **owner access**	7	⇒	1 1 1
			RWX
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



```
chmod 761 game
```

Attach a group to a file

```
chgrp G game
```