

# File Management

# File Concept

- A file is a named collection of related information that is recorded on secondary storage.
- Many different types of information may be stored in a file—source or executable programs, numeric or text data, photos, music, video, and so on.
- A file has a certain defined structure, which depends on its type.
- A **text file** is a sequence of characters organized into lines (and possibly pages).
- A **source file** is a sequence of functions, each of which is further organized as declarations followed by executable statements.
- An **executable file** is a series of code sections that the loader can bring into memory and execute.

# 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 file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – 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

# File Operations

- **Create**
- **Write** – at write pointer location
- **Read** – at read pointer location
- **Reposition within file - seek**
- **Delete**
- **Truncate** - erase the contents of a file but keep its attributes.
- ***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

# 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 processes closes it
  - Disk location of the file: cache of data access information
  - Access rights: per-process access mode information

# File Types – Name, Extension

| file type      | usual extension             | function   |
|----------------|-----------------------------|--|
| executable     | exe, com, bin<br>or none    | ready-to-run machine-<br>language program  |
| object         | obj, o                      | compiled, machine<br>language, not linked  |
| source code    | c, cc, java, pas,<br>asm, a | source code in various<br>languages  |
| batch          | bat, sh                     | commands to the command<br>interpreter   |
| text           | txt, doc                    | textual data, documents  |
| word processor | wp, tex, rtf,<br>doc        | various word-processor<br>formats  |
| library        | lib, a, so, dll             | libraries of routines for<br>programmers   |
| print or view  | ps, pdf, jpg                | ASCII or binary file in a<br>format for printing or<br>viewing                                 |
| archive        | arc, zip, tar               | related files grouped into<br>one file, sometimes com-<br>pressed, for archiving<br>or storage |
| multimedia     | mpeg, mov, rm,<br>mp3, avi  | binary file containing<br>audio or A/V information   |

# Access Methods

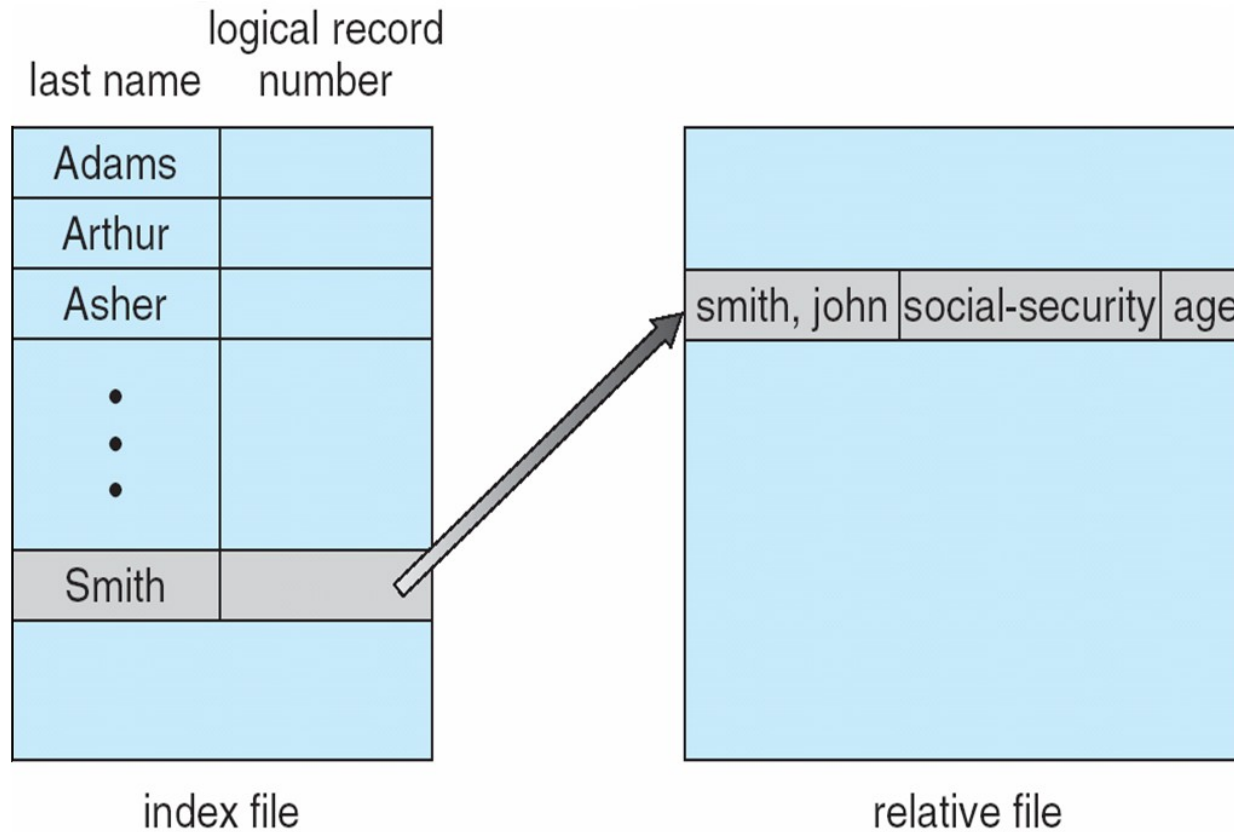
- **Sequential Access** - processed in order, one record after the other.
- **Direct Access** - The direct-access method is based on a disk model of a file, since disks allow random access to any file block.

# 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 Universal Produce Code (UPC code) plus record of data about that item)
- If too large, index (in memory) of the index (on disk).

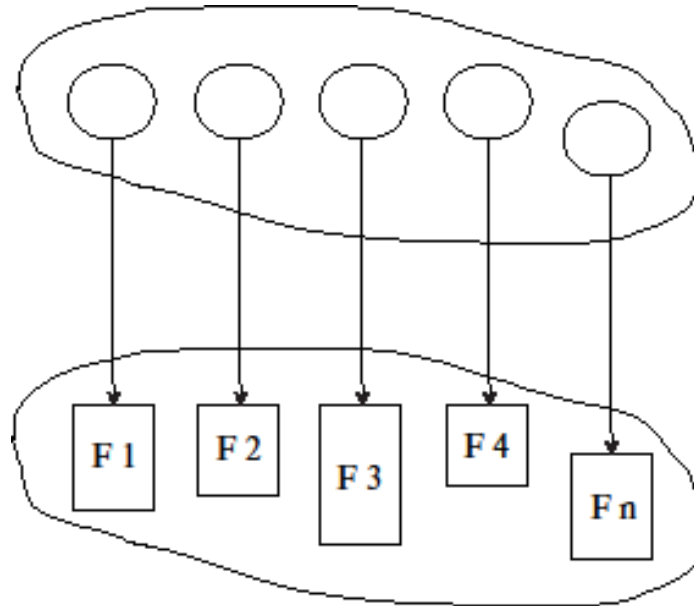


# Index and Relative Files



# Directory Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk

# Operations Performed on Directory

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

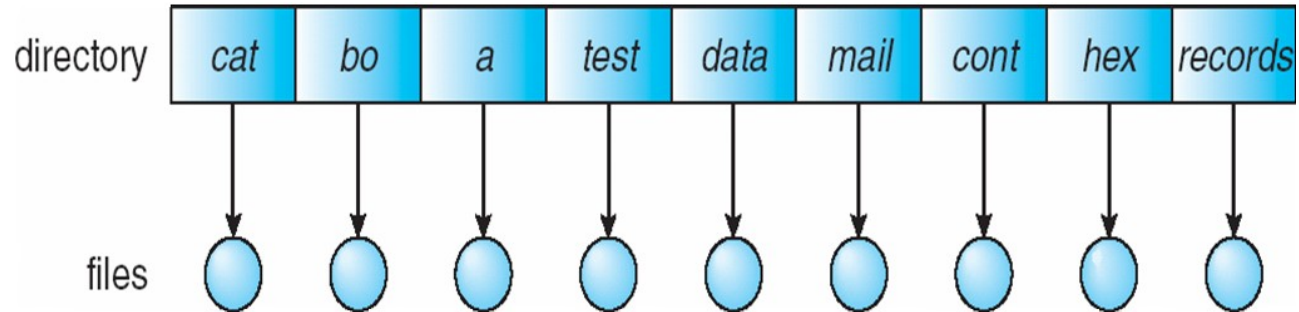
# Directory Organization

The directory is organized logically to obtain

- Efficiency – locating a file quickly
- 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

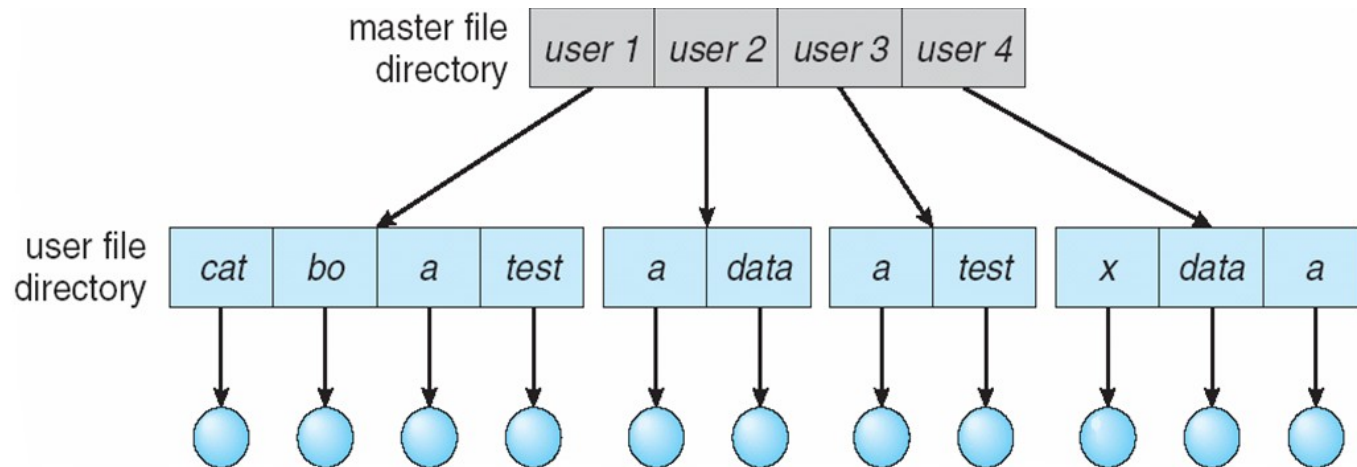
- A single directory for all users



- Naming problem
- Grouping problem

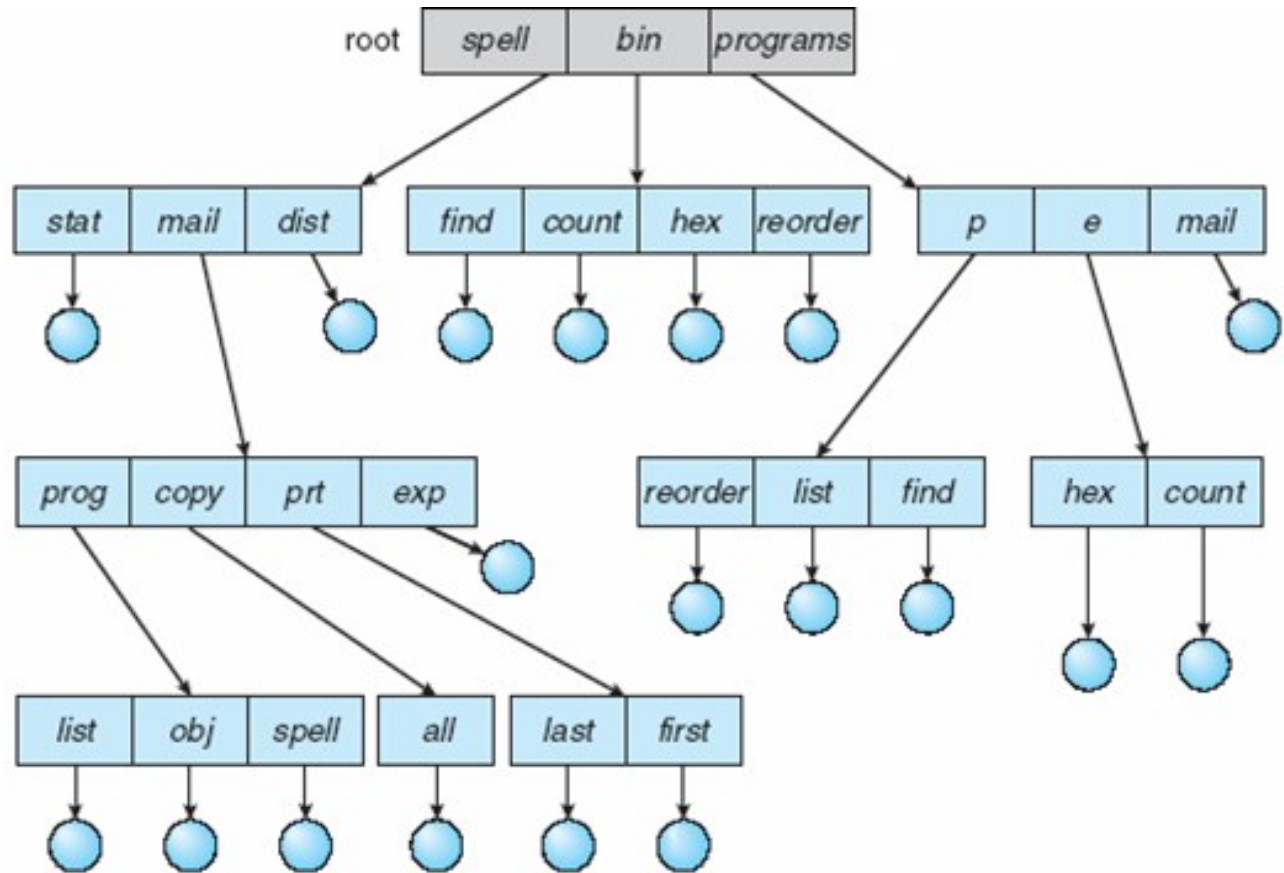
# Two-Level Directory

- Separate directory for each user



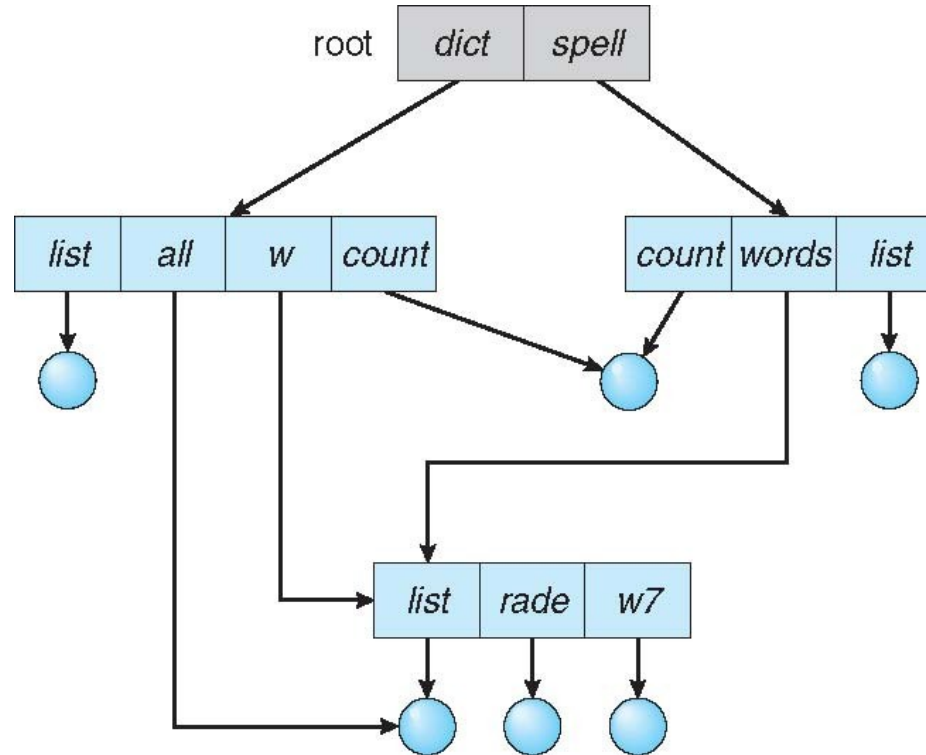
- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

# Tree-Structured Directories



# Acyclic-Graph Directories

- Have shared subdirectories and files





# Acyclic-Graph Directories

- Two different names (aliasing)
- If ***dict*** deletes ***w/list***  $\Rightarrow$  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

# 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
- 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

# File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Automatically, seamlessly using distributed file systems
  - Semi automatically via the world wide web
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - NFS is standard UNIX client-server file sharing protocol
  - CIFS is standard Windows protocol
  - Standard operating system file calls are translated into remote calls
- **Distributed Information Systems (distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

# Protection

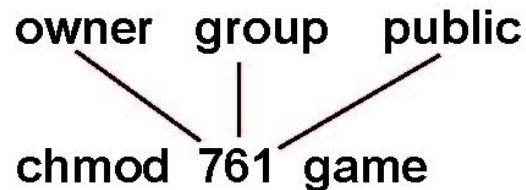
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

# Access Lists and Groups in Unix

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

|                         |   |     |       |
|-------------------------|---|-----|-------|
|                         |   | RWX |       |
| a) <b>owner access</b>  | 7 | ⇒   | 1 1 1 |
|                         |   | RWX |       |
| b) <b>group access</b>  | 6 | ⇒   | 1 1 0 |
|                         |   | RWX |       |
| c) <b>public access</b> | 1 | ⇒   | 0 0 1 |

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

**chgrp**      **G**      **game**

# Windows 7 Access-Control List Management

