Part III
Storage Management
Chapter 10: File-System Interface

Files

- A file is a named collection of related information that is recorded on secondary storage.
- The operating systems maps this logical storage unit to the physical view of information storage.
- A file may have the following characteristics
  - File Attributes
  - File Operations
  - File Types
  - File Structures
  - Internal Files

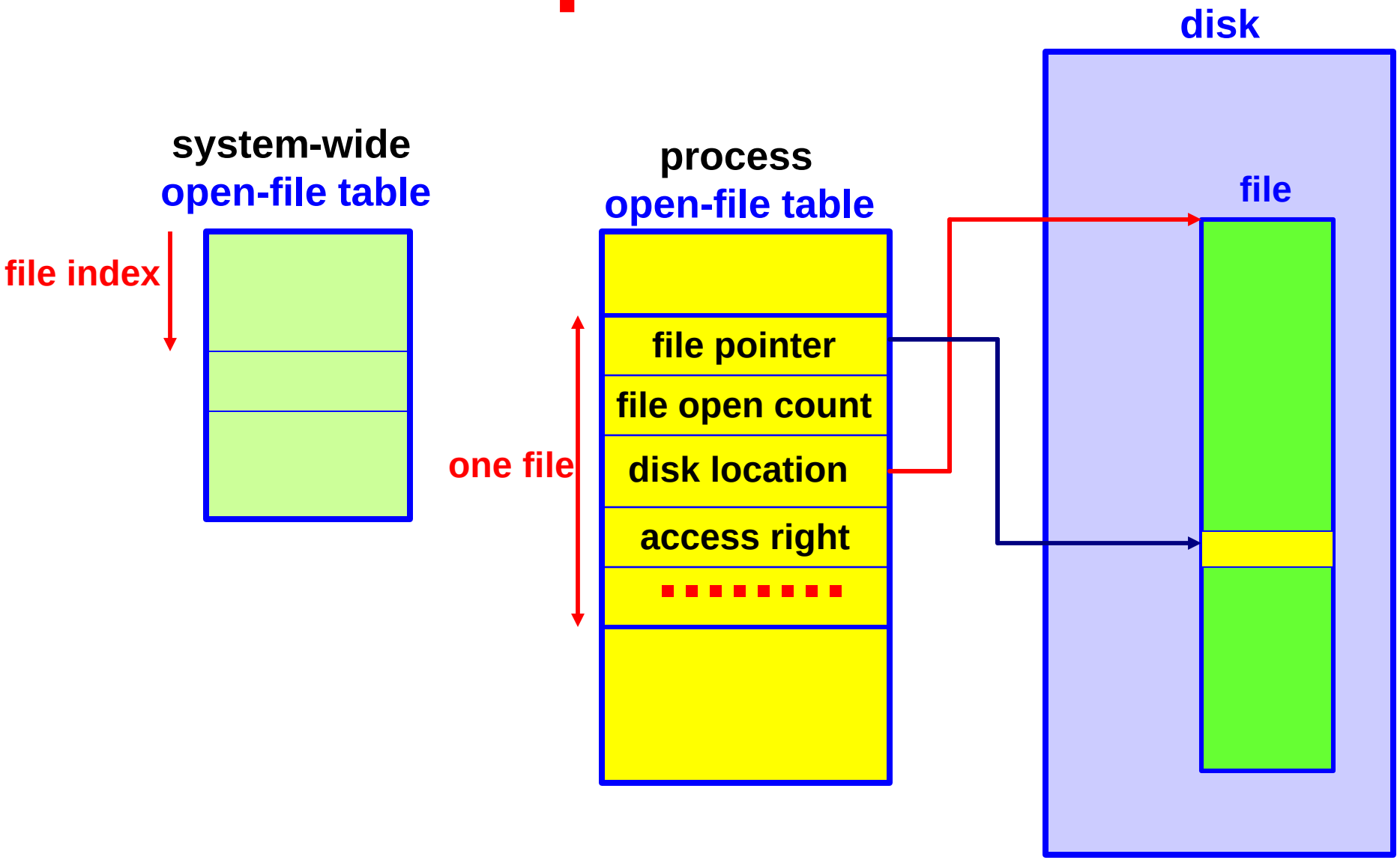
File Attributes

- **File Name: The symbolic name is perhaps the only human readable file attribute.**
- **Identifier: A unique number assigned to each file for identification purpose.**
- **File Type: Some systems recognize various file types. Windows is a good example.**
- **File Location: A pointer to a device to find a file.**
- **File Size: The current size of a file, or the maximum allowed size.**
- **File Protection: This is for access-control.**
- **File Date, Time, Owner, etc.**

File Operations: 1/2

- A file can be considered as an abstract data type that has data and accompanying operations.
- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file
- Other operations (*e.g.*, appending a file, renaming a file)

File Operations: 2/2



File Structure

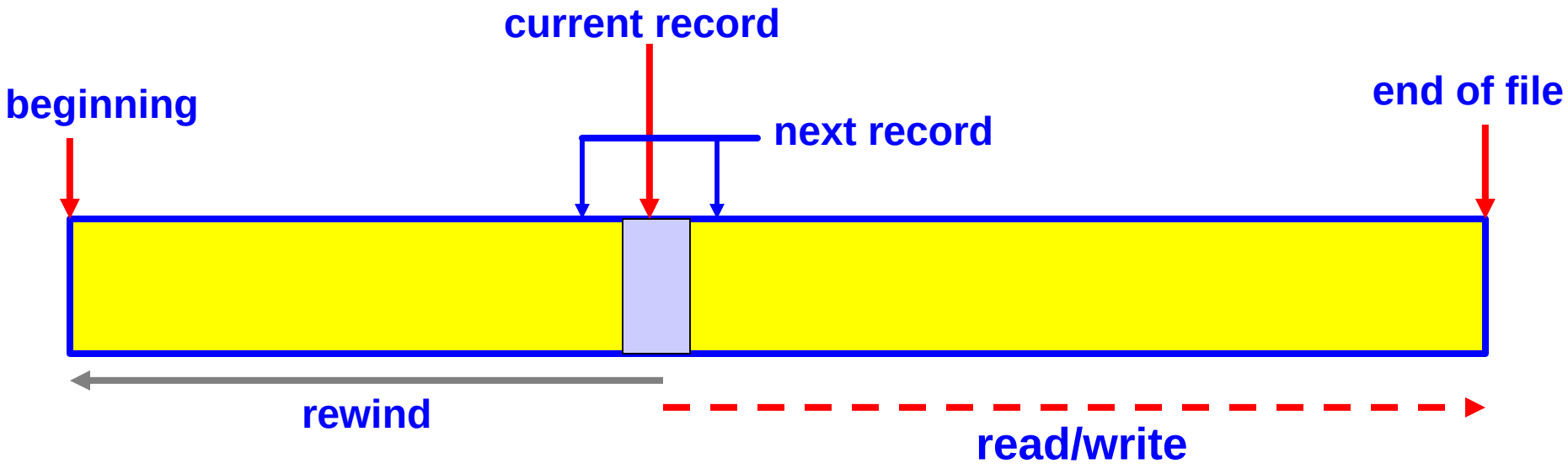
- **Some systems support specific file types that have special file structures.**
- **For example, files that contain binary executables.**
- **An operating system becomes more complex when more file types (*i.e.*, file structures) are supported.**
- **In general, the number of supported file types is kept to minimum.**

File Access Methods

- Access method: how a file be used.
- There are three popular ones:
  - Sequential access method for sequential files
  - Direct access method for direct files
  - Indexed access method for indexed files.

Sequential Access Method

- With the sequential access method, the file is processed in order, one record after the other.
- If *p* is the file pointer, the next record to be accessed is either *p*+1 or *p*-1 (*i.e.*, backspace).

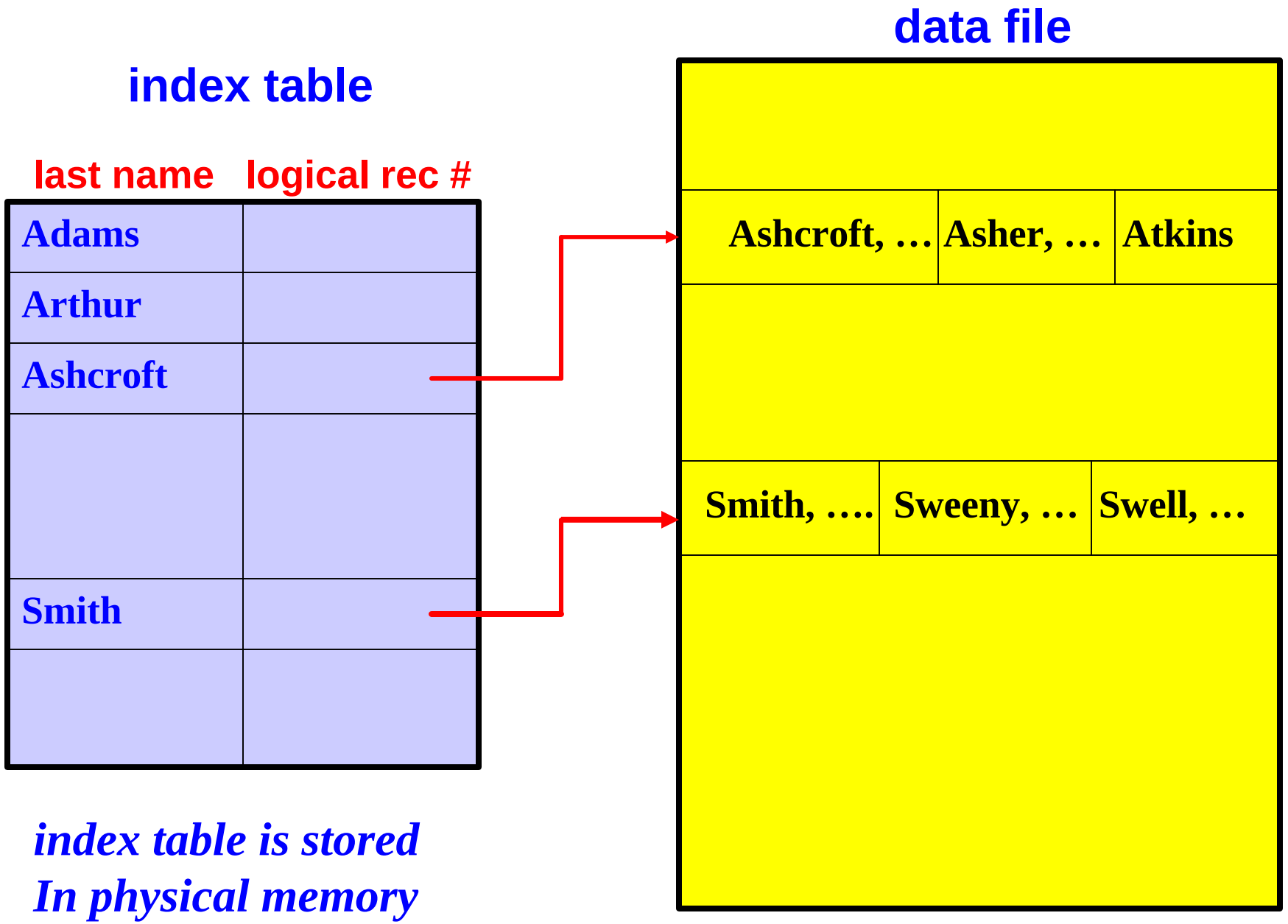


Direct Access Method

- A file is made up of fixed-length logical records.
- The direct access method uses record number to identify each record. For example, *read rec 0*, *write rec 100*, *seek rec 75*, etc.
- Some systems may use a key field to access a record (*e.g.*, *read rec "Age=24"* or *write rec "Name=Dow"*). This is usually achieved using hashing.
- Since records can be accessed in random order, direct access is also referred to as random access.
- Direct access method can simulate sequential access.

Indexed Access Method

- With the indexed access method, a file is sorted in ascending order based on a number of keys.
- Each disk block may contain a number of fixed-length logical records.
- An index table stores the keys of the first block in each block.
- We can search the index table to locate the block that contains the desired record. Then, search the block to find the desired record.
- This is exactly a one-level B-, B+ or B* tree.
- Multi-level index access method is also possible.


data file
index table
last name
logical rec #
Adams
Arthur
Ashcroft
Smith
Ashcroft, …
Asher, …
Atkins
Smith, ….
Sweeny, …
Swell, …
index table is stored
In physical memory

Directory Structure: 1/2

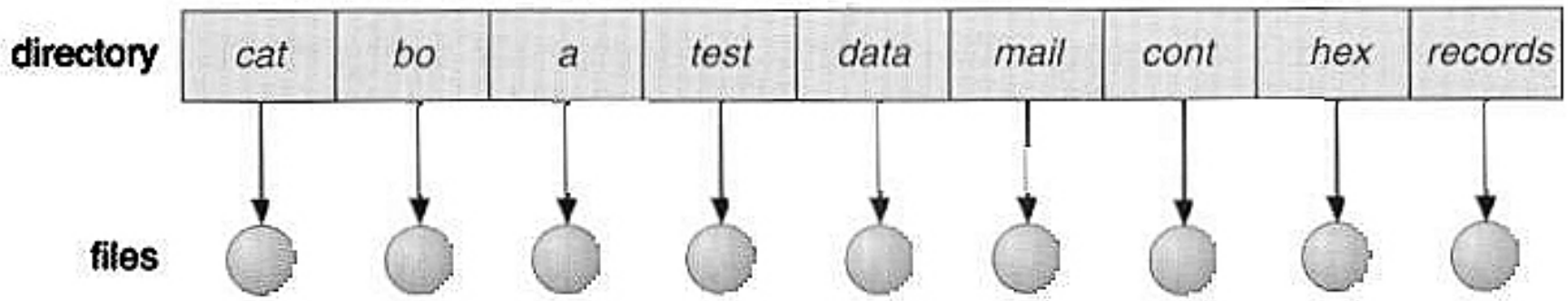
- A large volume disk may be partitioned into partitions, or mini disks, or volumes.
- Each partition contains information about files within it. This information is stored in entries of a device directory or volume table of content (VTOC).
- The device directory, or directory for short, stores the name, location, size, type, access method, etc of each file.
- Operations perform on directory: search for a file, create a file, delete a file, rename a file, traverse the file system, etc.

Directory Structure: 2/2

- There are five commonly used directory structures:
  - Single-Level Directory
  - Two-Level Directory
  - Tree-Structure Directories
  - Acyclic-Graph Directories
  - General Graph Directories

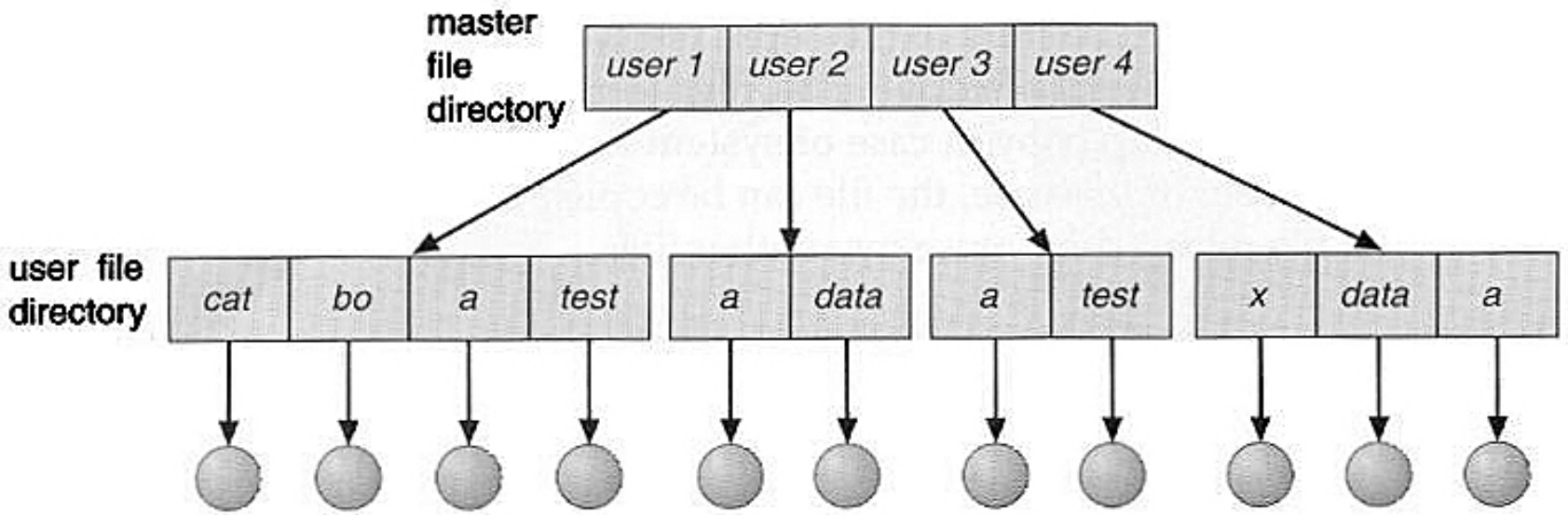
Single-Level Directory

- All files are contained in the same directory.
- It is difficult to maintain file name uniqueness.
- CP/M-80 and early version of MS-DOS use this directory structure.



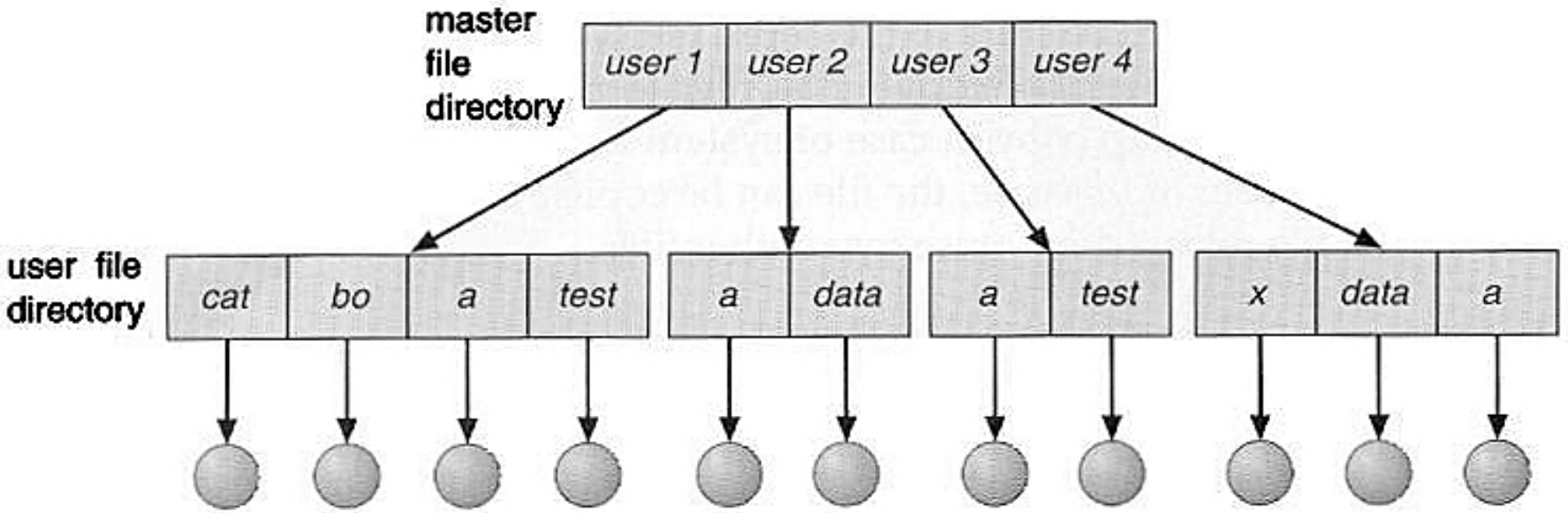
Two-Level Directory: 1/2

- This is an extension of the single-level directory for multi-user system.
- Each user has his/her user file directory. The system's master file directory is searched for the user directory when a user job starts.
- Early CP/M-80 multi-user systems use this structure.



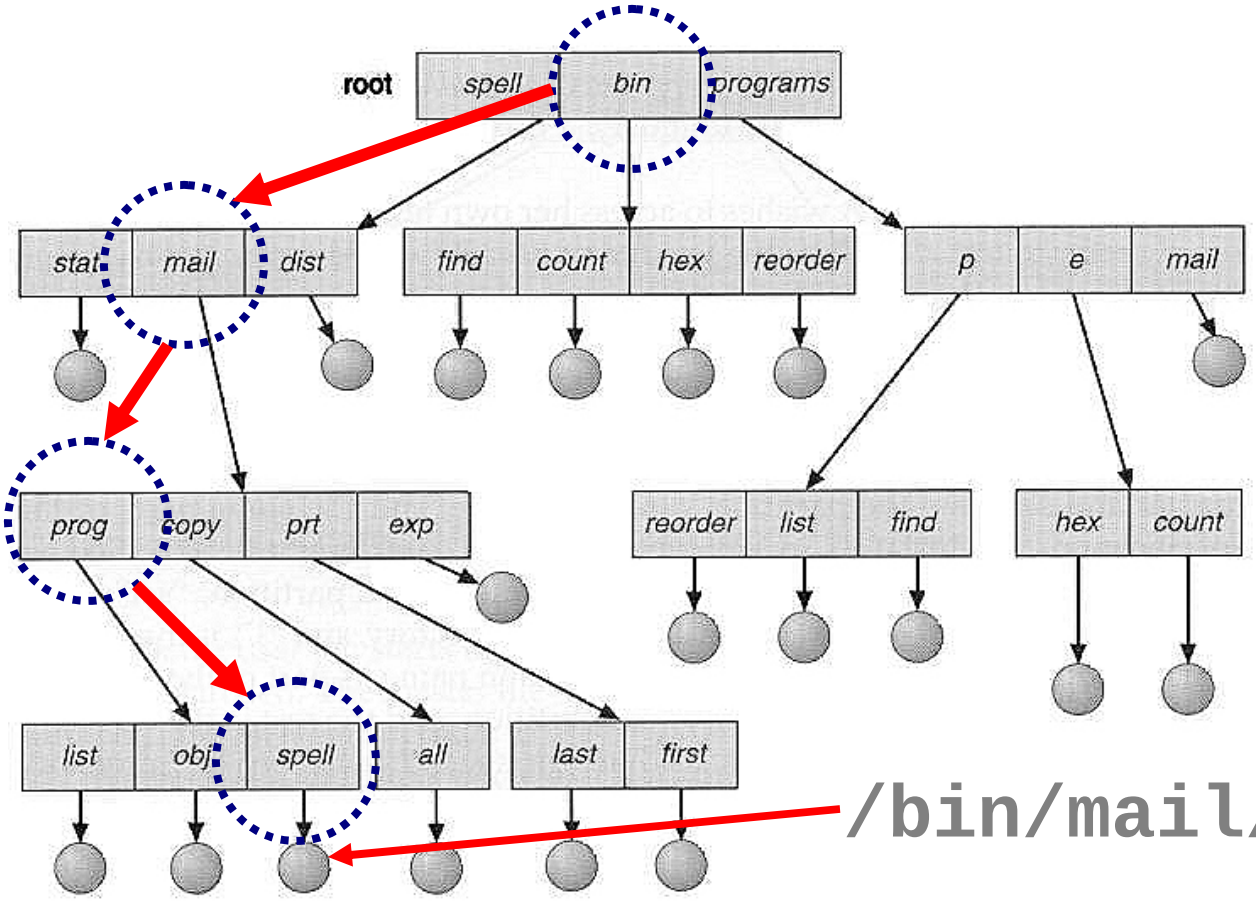
Two-Level Directory: 2/2

- To locate a file, path name is used. For example, `/user2/bo` is the file `bo` of `user 2`.
- Different systems use different path names. For example, under MS-DOS it may be `C:\user2\bo`.
- The directory of a special user, say `user 0`, may contain all system files.

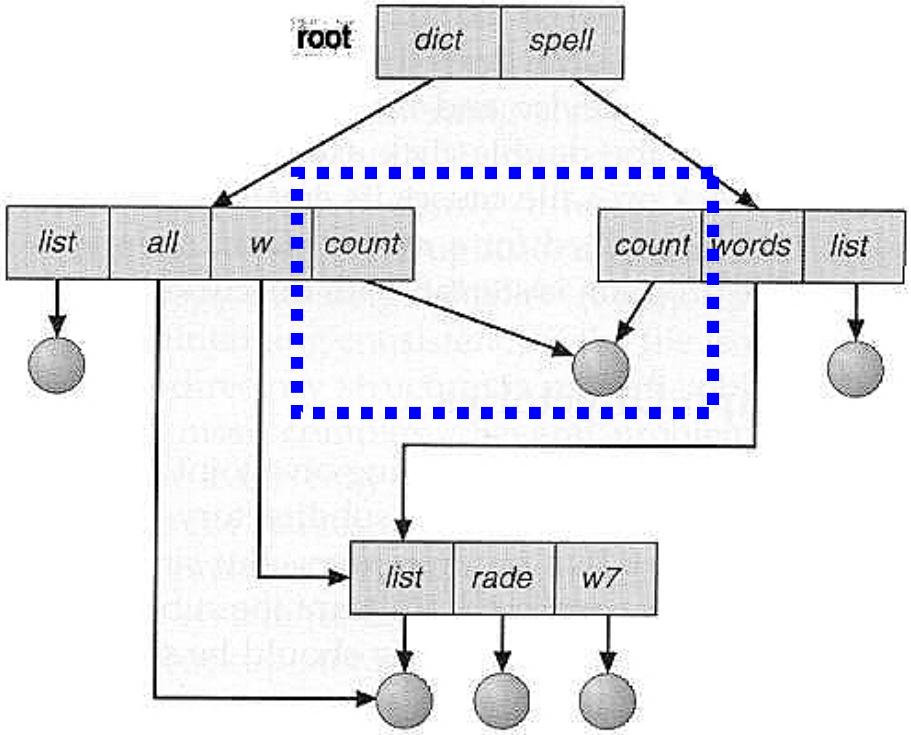


Tree-Structured Directory

- Each directory or subdirectory contains files and subdirectories, and forms a tree.
- Directories are special files.



Acyclic-Graph Directory: 1/2



file `count` is shared by directories `dict` and `spell`

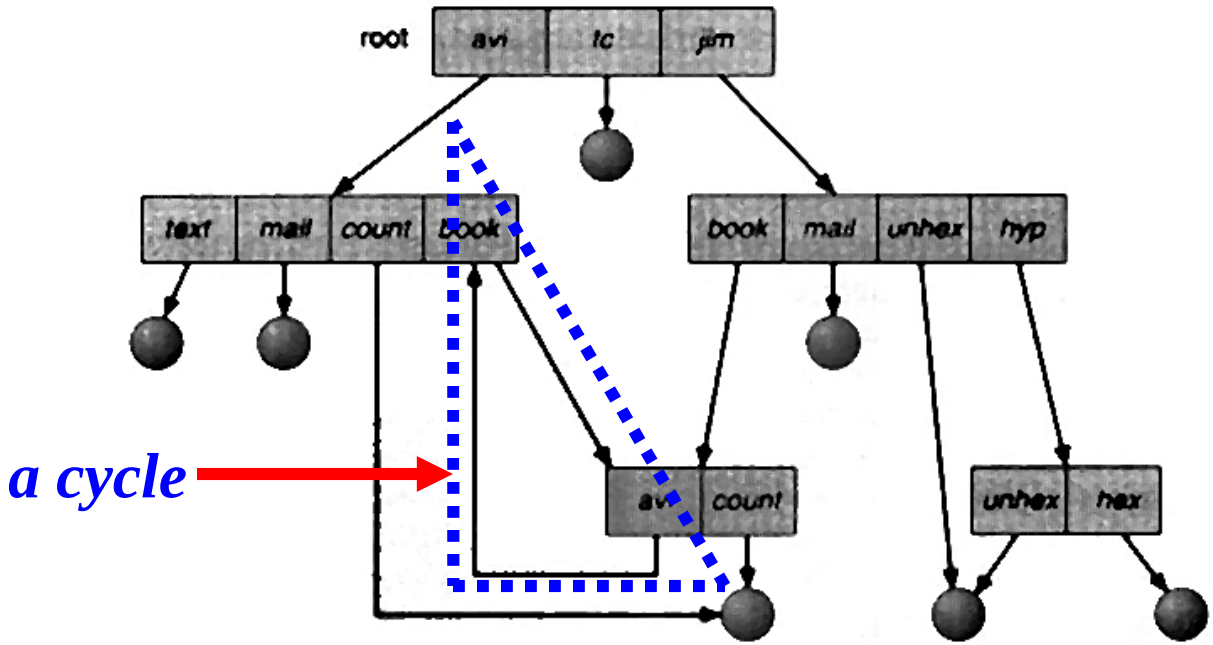
- This type of directories allows a file/directory to be shared by multiple directories.
- This is different from two copies of the same file or directory.
- An acyclic-graph directory is more flexible than a simple tree structure. However, it is more complex.

Acyclic-Graph Directory: 2/2

- Since a file have multiple absolute path names, how do we calculate file system statistics or do backup? Would the same file be duplicated multiple times?
- How do we delete a file?
  - If sharing is implemented with symbolic links, we only delete the link if we have a list of links to the file. The file is removed when the list is empty.
  - Or, we remove the file and keep the links. When the file is accessed again, a message is given and the link is removed.
  - Or, we can maintain a reference count for each shared file. The file is removed when the count is zero.

General Graph Directory: 1/2

- It is easy to traverse the directories of a tree or an acyclic directory system.
- However, if links are added arbitrarily, the directory graph becomes arbitrary and may contain cycles.
- *How do we search for a file?*



General Graph Directory: 2/2

- How do we delete a file? We can use reference count!
  - In a cycle, due to self-reference, the reference count may be non-zero even when it is no longer possible to refer to a file or directory.
  - Thus, garbage collection may needed. A garbage collector traverses the directory and marks files and directories that can be accessed.
  - A second round removes those inaccessible items.
- To avoid this time-consuming task, a system can check if a cycle may occur when a link is made. How? You should know!

File Sharing

- When a file is shared by multiple users, how can we ensure its consistency?
- If multiple users are writing to the file, should all of the writers be allowed to write?
- Or, should the operating system protect the user actions from each other?
- This is the file consistency semantics.

File Consistency Semantics

- **Consistency semantics is a characterization of the system that specifies the semantics of multiple users accessing a *shared* file *simultaneously*.**
- **Consistency semantics is an important criterion for evaluating any file system that supports file sharing.**
- **There are three commonly used semantics**
  - **Unix semantics**
  - **Session Semantics**
  - **Immutable-Shared-Files Semantics**
- **A file session consists all file access between `open()` and `close()`.**

Unix Semantics

- Writes to an open file by a user are visible immediately to other users have the file open at the same time.
- All users share the file pointer. Thus, advancing the file pointer by one user affects all sharing users.
- A file has a single image that interleaves all accesses, regardless of their origin.

Session Semantics

- Writes to an open file by a user are not visible immediately to other users that have the same file open simultaneously.
- Once a file is closed, the changes made to it are visible only in sessions started later.
- Already-open instances of the file do not affect these changes.
  - A file may be associated temporarily with several and possible different images at the same time.
  - Multiple users are allowed to perform both read and write concurrently on their image of the file without delay.
- The Andrew File System (AFS) uses this semantics.

Immutable-Shared-Files Semantics

- Once a file is declared as shared by its creator, it **cannot** be modified.
- An immutable file has two important properties:
  - Its name may not be **used**
  - Its content may not be **altered**
- Thus, the name of an immutable file indicates that the contents of the file is **fixed** – a constant rather than a variable.
- The implementation of these semantics in a distributed system is simple, since sharing is disciplined (*i.e.*, **read-only**).

File Protection

- We can keep files safe from physical damage (*i.e.*, reliability) and improper access (*i.e.*, protection).
- *Reliability* is generally provided by backup.
- The need for *file protection* is a direct result of the ability to access files.
- Access control may be a complete protection by denying access. Or, the access may be controlled.

File Protection: Types of Access

- Access control may be implemented by limiting the types of file access that can be made.
- The types of access may be
  - **Read**: read from the file
  - **Write**: write or rewrite the file
  - **Execute**: load the file into memory and execute it
  - **Append**: write new info at the end of a file
  - **Delete**: delete a file
  - **List**: list the name and attributes of the file

File Protection: Access Control: 1/4

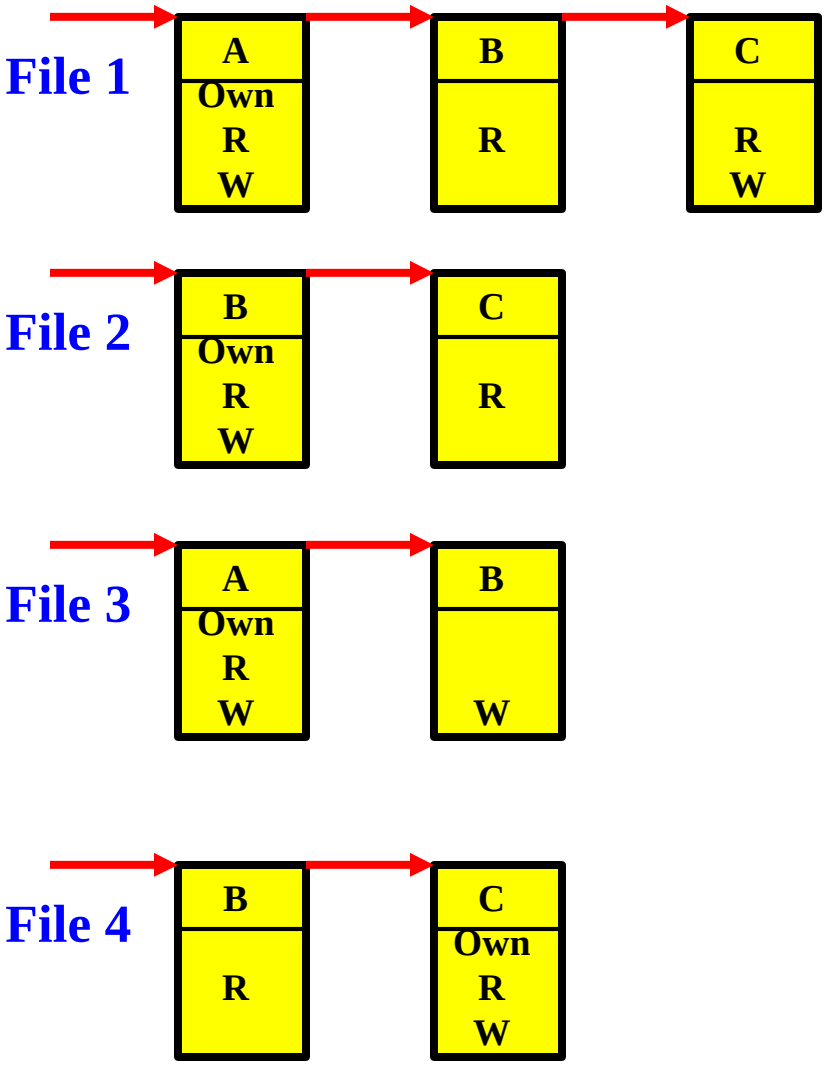
- The most commonly used approach is to make the access dependent on the identity of the user.
- Each file and directory is associated with an access matrix specifying the user name and the types of permitted access.
- When a user makes a request to access a file or a directory, his/her identity is compared against the information stored in the access matrix.

File Protection: Access Control: 2/4

Access Matrix

	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry debit

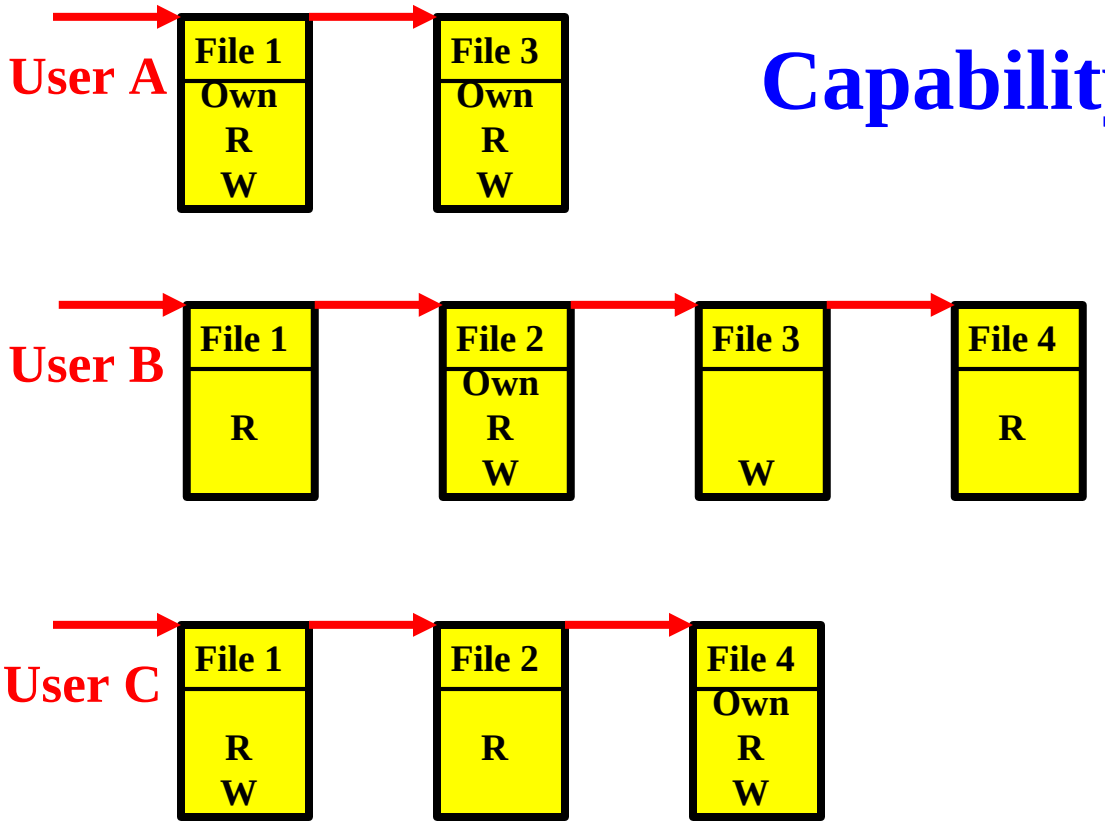
File Protection: Access Control: 3/4



Access-control Lists

- In practice, the access matrix is sparse.
- The matrix can be decomposed into columns (files), yielding access-control lists (ACL)
- However, this list can be very long!

File Protection: Access Control: 4/4



Capability Lists

- Decomposition by rows (users) yields capability tickets.
- Each user has a number ticket for file/directory access.