Operating Systems

BITS Pilani
K K Birla Goa Campus

Dr. Lucy J. Gudino
Dept. of CS and IS

File System Interface

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

Introduction

- 3 essential requirements for long-term information storage
  - It must be possible to store a very large amount of information.
  - The information must survive the termination of the process using it.
  - Multiple processes must be able to access the information concurrently.
- Solution : files
- Information stored in files must **be persistent**, that is, not be affected by process creation and termination.
- Files are managed by OS
- part of the operating system dealing with files is known as the **file system or file management system**

File Concept

- File : is a resource for storing the information
- is a named collection of related information that is recorded on secondary storage
- have life outside individual application
- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

Contd…

- Types of information : source programs, object programs, executable programs, numeric data, text, payroll records, graphic images, sound recordings, and so on

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system (non-human readable name for the file)
- **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**

Field	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	Id of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

File Operations

- File is an **abstract data type**
- **File Operations**
  - **Create**
  - **Write**
  - **Read**
  - **Reposition within file**
  - **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 Operations

- Creating a file:
  - Find space for the file
  - make entry in the directory
- Writing a file
  - System call with file name and information to be written
  - Uses write pointer
  - update write pointer after writing
- Reading a file:
  - System call with file name and where the information to be put
  - Uses read pointer
  - Update the read pointer after reading
- Instead of having two pointers use single pointer known as “ current file position pointer”

File Operations

- Repositioning within a file
  - also known as seek operation
  - current file position pointer is repositioned to a given value
- Deleting a file
  - system call with file name as parameter
  - searches in the directory for the file
  - if found release the space allotted to it and erase the directory entry
- Truncating a file:
  - erases the contents of the file
  - attributes remains unchanged

Other file operations

- Appending: copying new information to the end of existing file
- Renaming a file
- copy old file to new file
  - create new file
  - read old file
  - write on to new file attributes
- Operations related to
  - setting attributes
  - getting attributes

Contd…

- Some systems use implicit open() and close() system calls
- Importance of **open-file table**
- open() system call
  - file name as parameter
  - Searches directory
  - if found copies directory entry into open-file table
  - can also accept access mode information such as read only, read write append-only and so on.
  - access mode parameter is compared with file permissions. If satisfied, file is opened.
  - returns a pointer to the entry in the open-file table
- close() system call

Contd…

- Several pieces of data are needed to manage 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

Open File Locking

- Provided by some operating systems and file systems
- Mediates access to a file
- File locking mechanisms can be Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested
    - once a process acquires an exclusive lock, the operating system will prevent any other process from accessing the locked file.
  - **Advisory** – processes can find status of locks and decide what to do

File naming

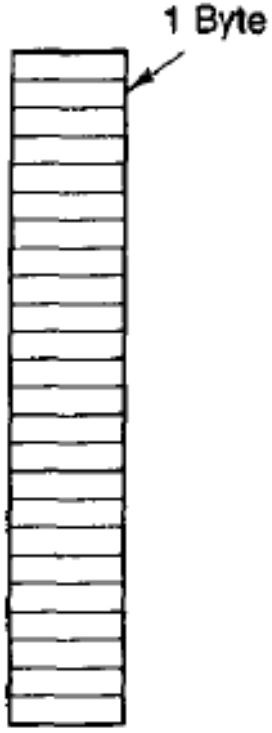
- Files are an abstraction mechanism.
- Name is given when file is created by a process,
- On process termination, the file continues to exist
- Alpha numeric file names : 8 to 256 characters.
- Case sensitive file name. EX: MS-DOS and UNIX
- Two part file names and separated by a period ”.”
  - Primary name
  - Secondary name
- In some cases, the file extensions ace just conventions and are not enforced in any way.

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-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, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	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, mp3, avi	binary file containing audio or A/V information

File Structure or organization

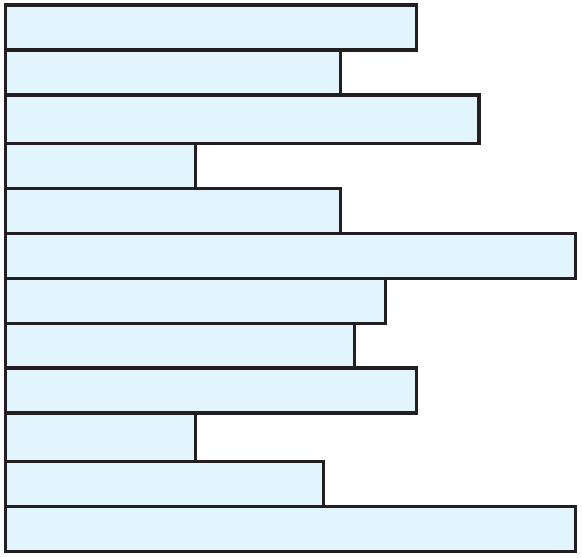
- None – Unstructured
  - sequence of words, bytes
  - Meaning determined by the programmer or user or application
  - provides maximum flexibility
  - Ex: Unix



Text:Tenanbaum

File Structure - Pile

- Data are collected in the order in which they arrive
- Each record contains burst of data
- least complicated

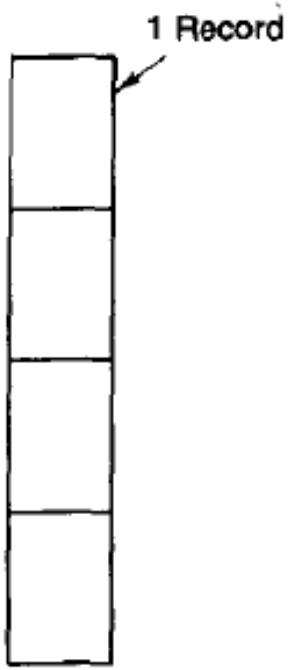


Variable-length records
Variable set of fields
Chronological order

Pile file

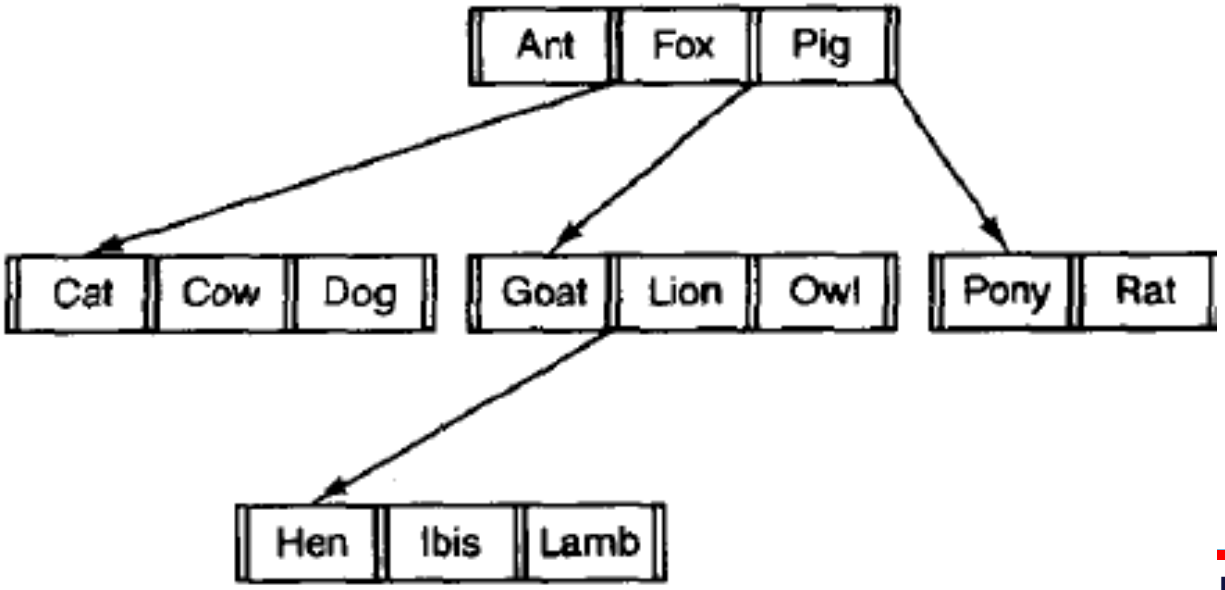
File Structure - Record

- Simple record structure
  - is a sequence of fixed-length records, each wit some internal structure
  - read and write operation – one record at a time



File Structure - Tree

- Tree structure:
  - contains trees of records
  - length of the records may be fixed or variable
  - each containing a key field in a fixed position in the record.
  - tree is sorted based on key field



Access Methods

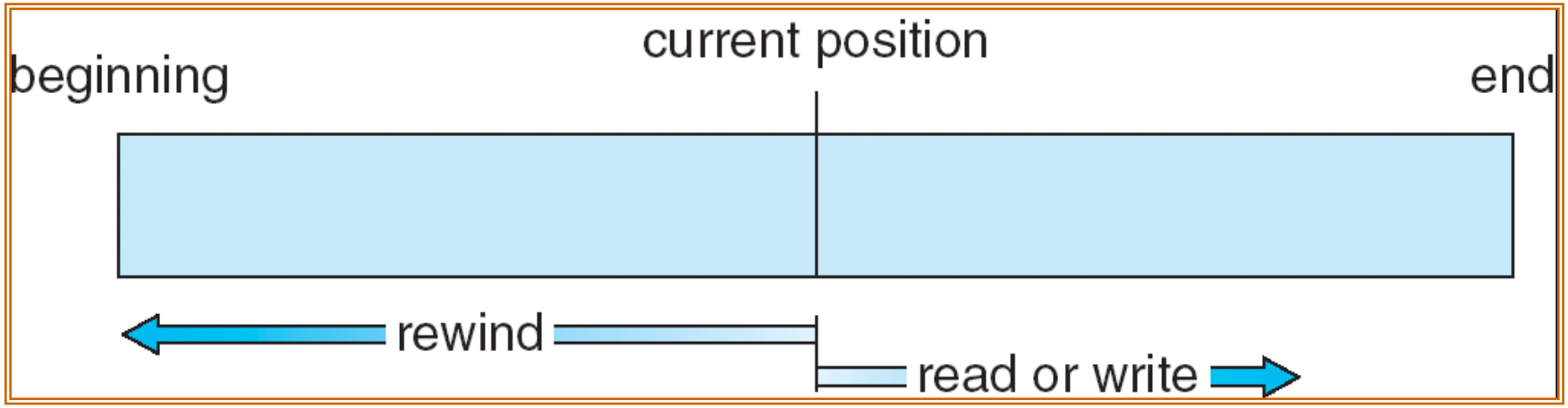
- Three access methods
  - Sequential access methods
  - Direct access methods
  - Keyed access method

Access Methods: Sequential

innovate achieve lead

- In order reading of bytes or records
- No skipping or reading out of order
- Example: editors and compilers
- Read operation : read next
  - reads the next portion of the file and automatically advances a file pointer
- Write operation: write next
  - appends to the end of the file and advances to the end of the newly written material (the new end of file)
- Rewind

Sequential-access File



Access Methods: Direct Access

- Also known as relative accessing or random accessing
- Can access any byte in the file directly, without accessing any of its predecessors
  - Example: accessing database record 12
- Operations:
  - read *n*
  - write *n*
  - position to *n*

    read next

    write next

Access methods – Contd…

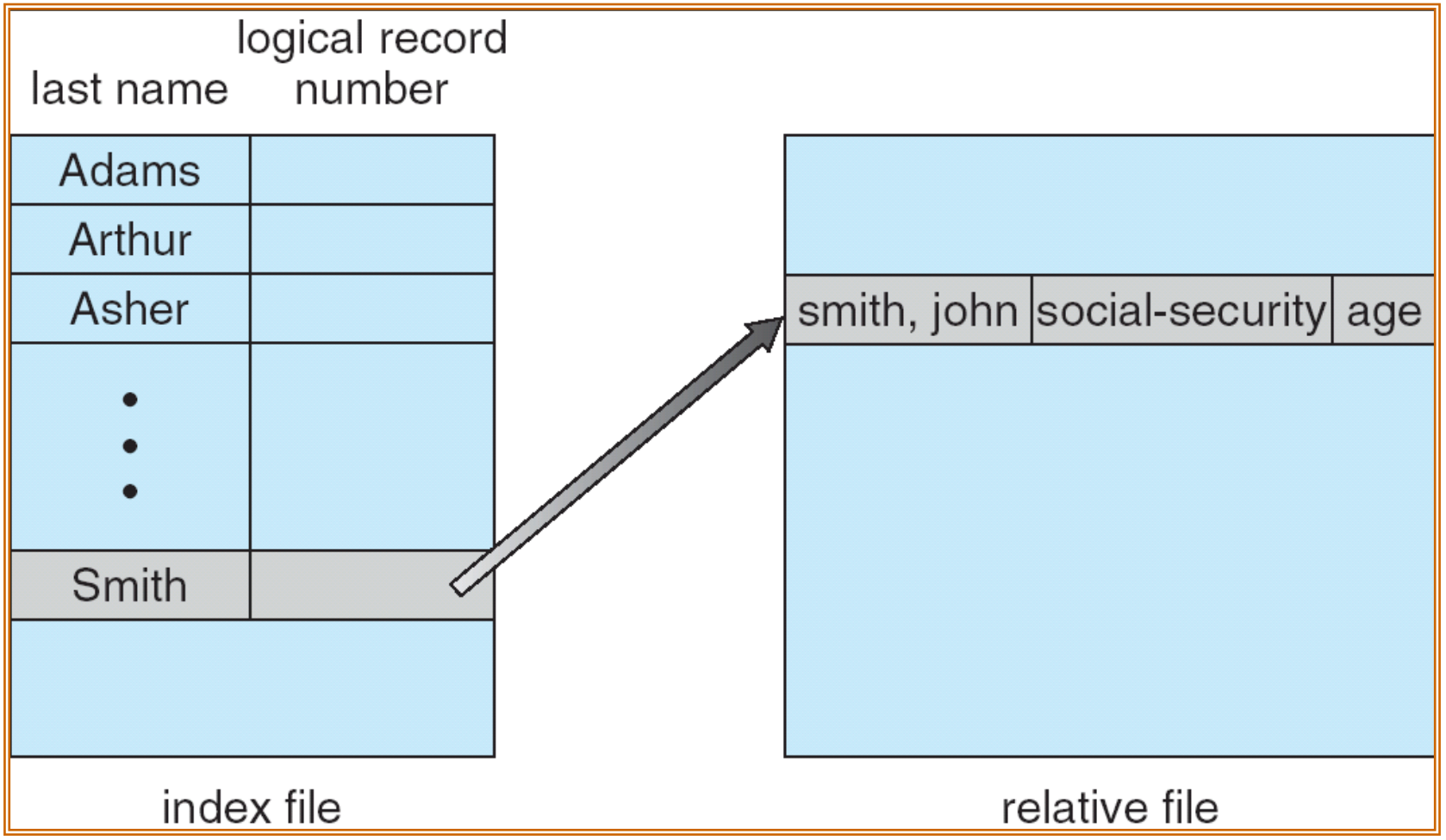
- access method : sequential or direct or both
- Some systems require that a file be defined as sequential or direct when it is created

Simulation of Sequential Access on a Direct-access File

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Other Access Methods: Keyed Access

- Can access a byte based on a key value
- Example: database search, dictionary
- OS does not support keyed access
  - User program must determine the address from the key, then use random access (provided by the OS) into the file


last name
logical record number
Adams
Arthur
Asher
Smith
smith, john
social-security
age
index file
relative file

File Management System

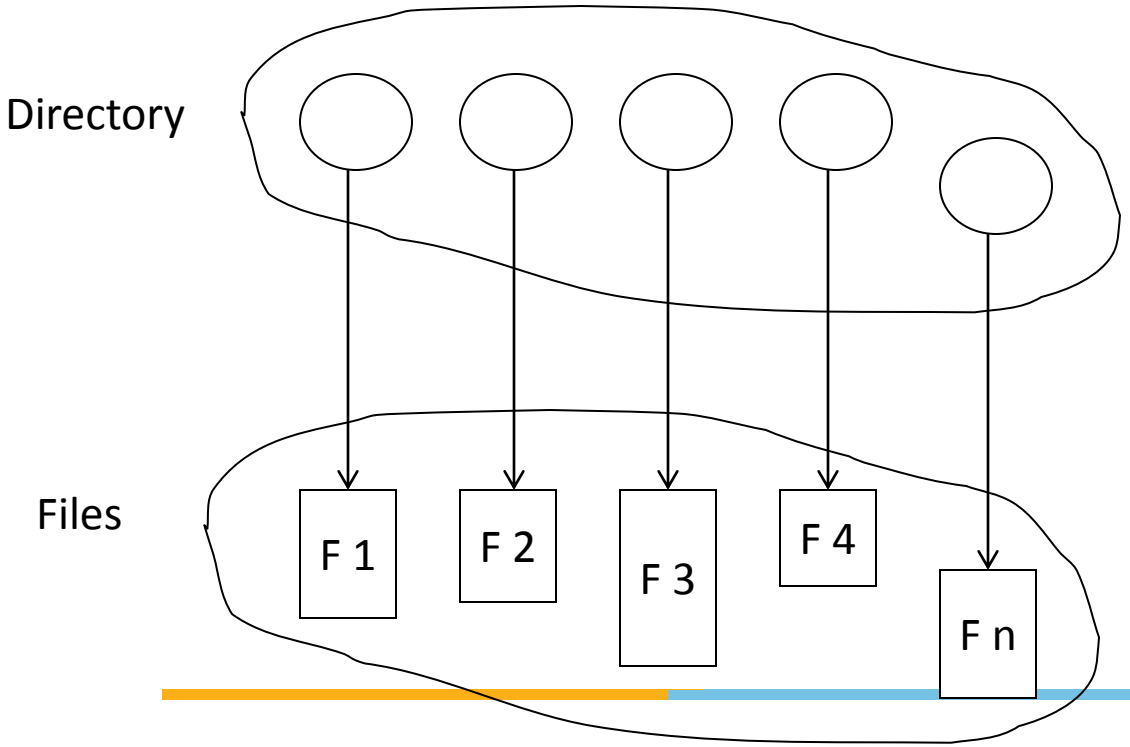
- is that set of system software that provides services to users and applications in the use of files.
- can access file through file management system
- Objectives:
  - To meet the data management needs and requirements of the user, which include storage of data and the ability to perform the aforementioned operations
  - To guarantee, to the extent possible, that the data in the file are valid
  - To optimize performance, both from the system point of view in terms of overall throughput and from the user's point of view in terms of response time
  - To provide I/O support for a variety of storage device types
  - To minimize or eliminate the potential for lost or destroyed data
  - To provide a standardized set of I/O interface routines to user processes
  - To provide I/O support for multiple users, in the case of multiple-user systems

Contd…

- Requirements
  1. Each user should be able to create, delete, read, write, and modify files.
  2. Each user may have controlled access to other users' files.
  3. Each user may control what types of accesses are allowed to the user's files.
  4. Each user should be able to restructure the user's files in a form appropriate to the problem.
  5. Each user should be able to move data between files.
  6. Each user should be able to back up and recover the user's files in case of damage.
  7. Each user should be able to access his or her files by name rather than by numeric identifier.

Directory Structure

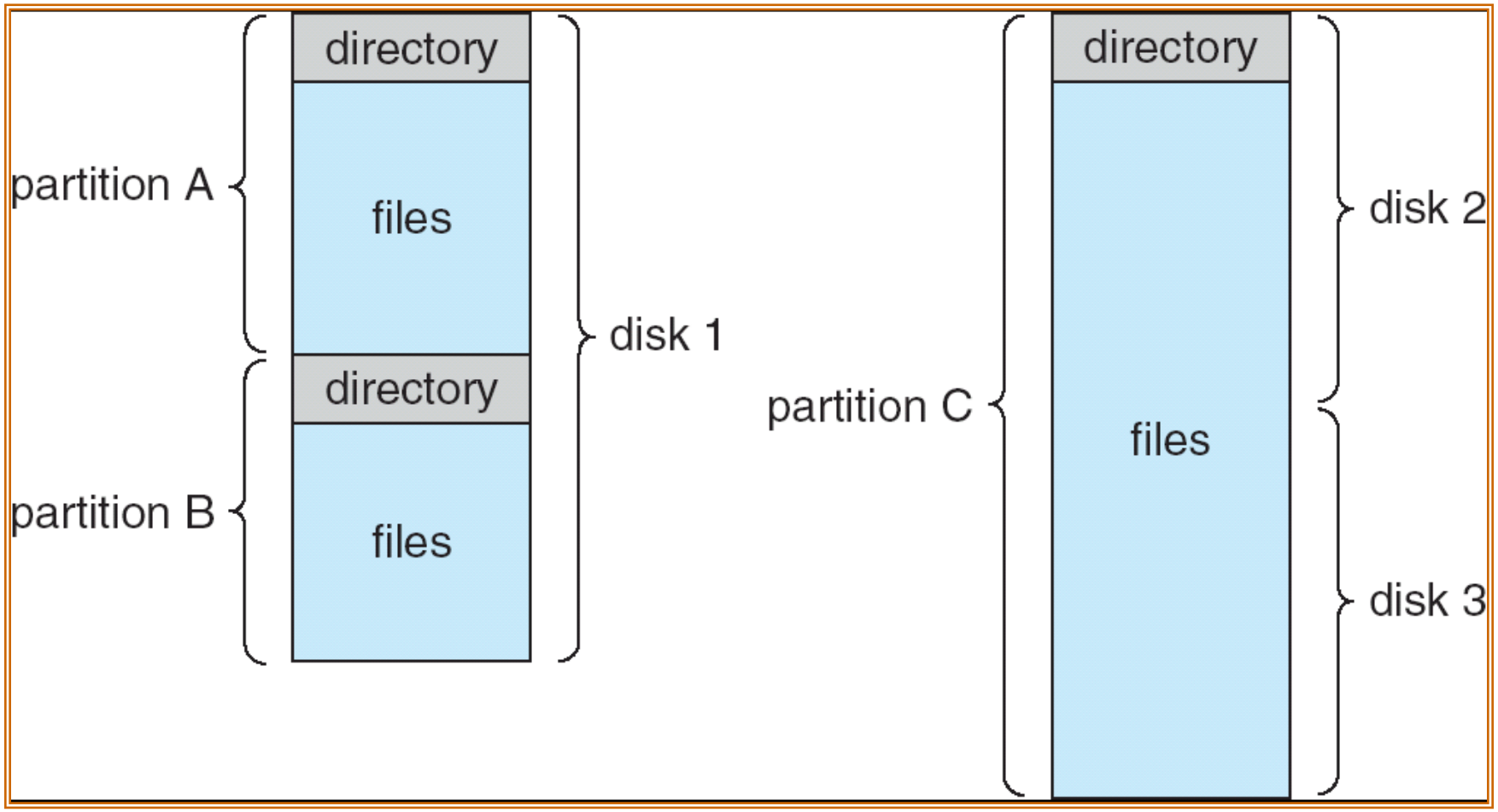
- Used to keep track of files
- are themselves files
- A collection of nodes containing information about all files
- Both the directory structure and the files reside on disk



Information Elements of a File Directory

	Basic Information
File Name	Name as chosen by creator (user or program). Must be unique within a specific directory.
File Type	For example: text, binary, load module, etc.
File Organization	For systems that support different organizations
	Address Information
Volume	Indicates device on which file is stored
Starting Address	Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
Size Used	Current size of the file in bytes, words, or blocks
Size Allocated	The maximum size of the file
	Access Control Information
Owner	User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges.
Access Information	A simple version of this element would include the user's name and password for each authorized user.
Permitted Actions	Controls reading, writing, executing, transmitting over a network
	Usage Information
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
Current Usage	Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, and whether the file has been updated in main memory but not yet on disk

A Typical File-system Organization



Operations Performed on Directory

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

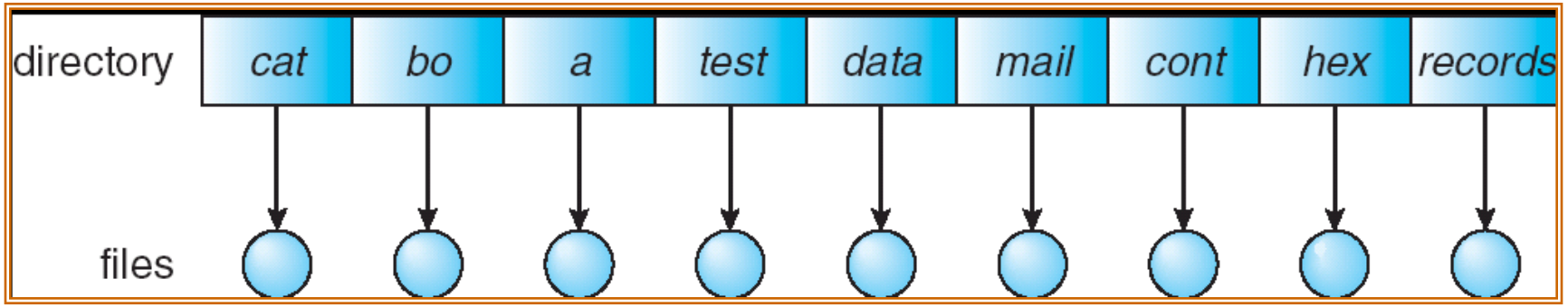
Organize the Directory (Logically) to Obtain

- Efficiency – 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

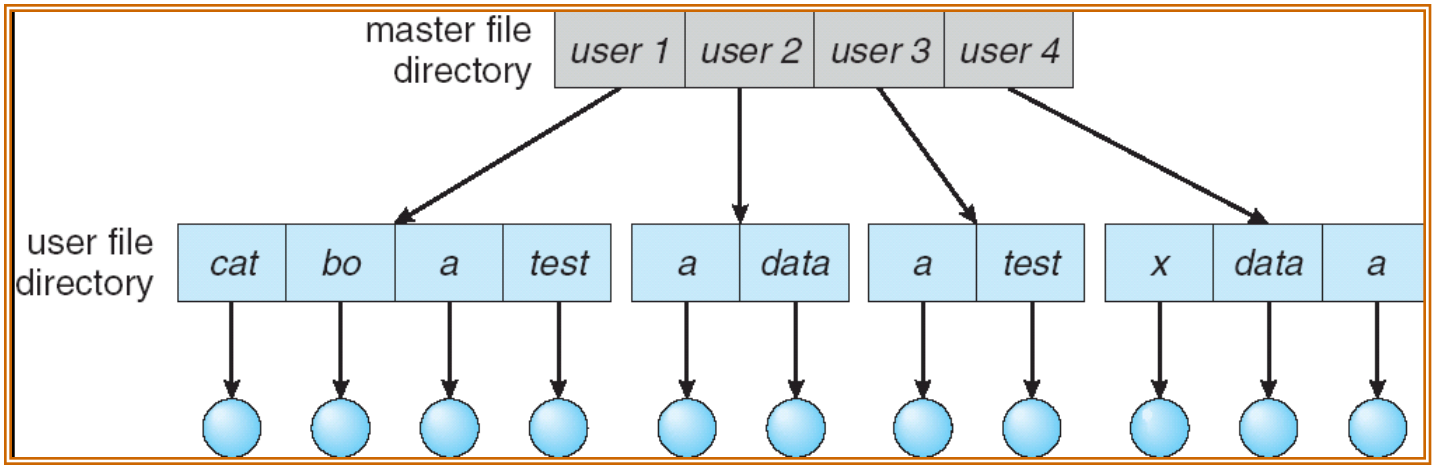
- The simplest solution:: A single-level directory with file entries for all users contained in the same directory.
- Advantages:
  - Easy to support and understand.
- Disadvantages:
  - Requires unique file names unique file names unique file names {the naming problem}.
  - No natural system for keeping track of file names {the grouping problem}.

Single-Level Directory



Two-Level Directory

- Standard solution: a separate directory for each user.
- The system's Master File Directory (MFD) has pointers to individual User File Directories
- File names default to localized UFD for all operations.



Contd…

- Advantages
  - Solves the name-collision problem.
  - Isolates users from one another a form of protection.
  - Efficient searching.
- Disadvantages
  - No logical grouping capability (other than by user).

Contd… Path name

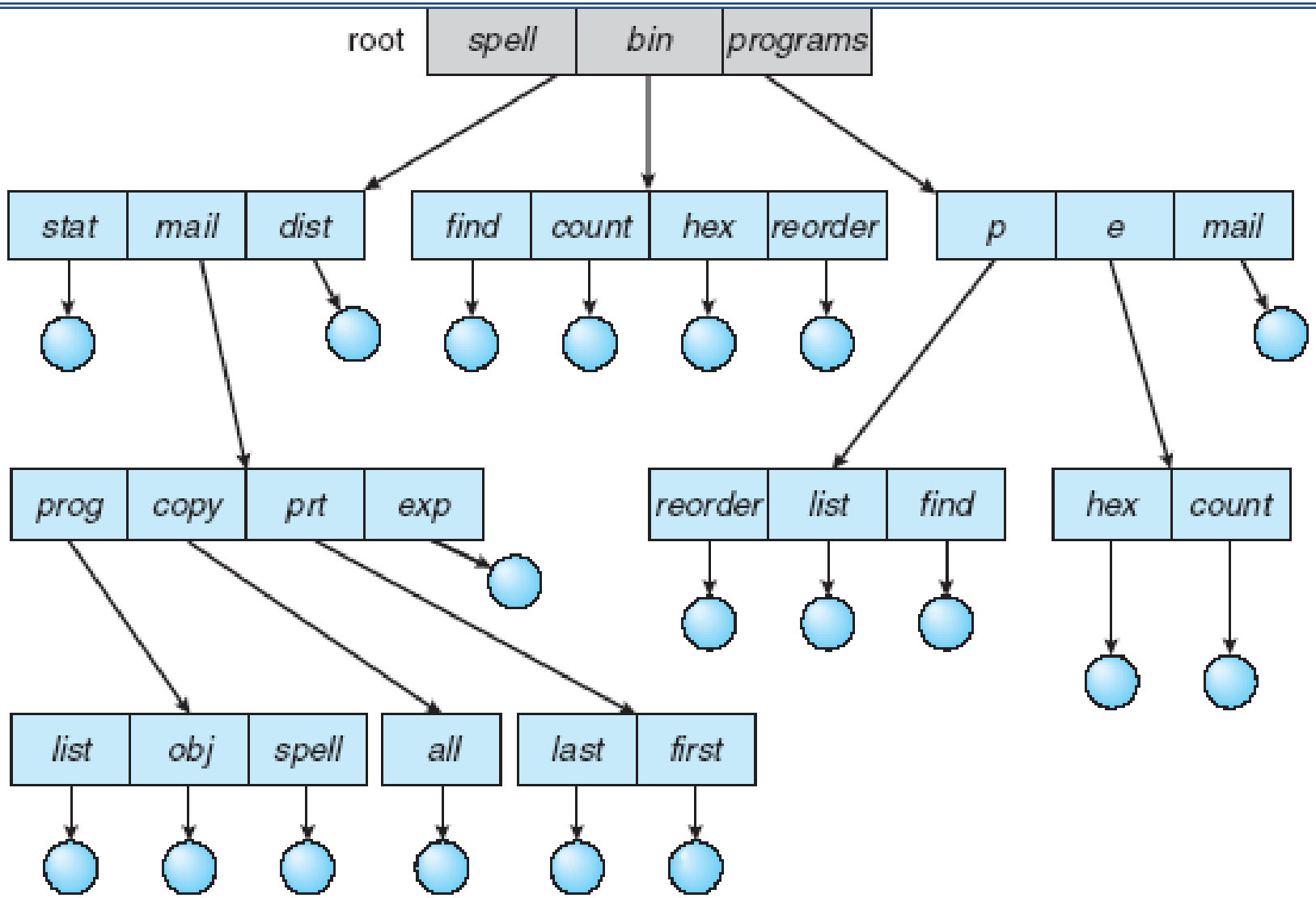
- If a user can access another user's files, the concept of path name path name path name is needed.
- In two-level directory, this tree structure has MFD as root of path through UFD to user file name at leaf.
- path name :: username + filename
- Standard syntax -- /user/file.txt

Tree-Structured Directories

innovate achieve lead

- Allows user to create subdirectories
- Directory:
  - Becomes simply another file
  - Contains a set of files or subdirectories.
  - All directories have the same internal format.
  - One bit in directory entry defines entry as file or directory.
  - Special commands are used to create and delete directories.

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Advantages :
  - 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

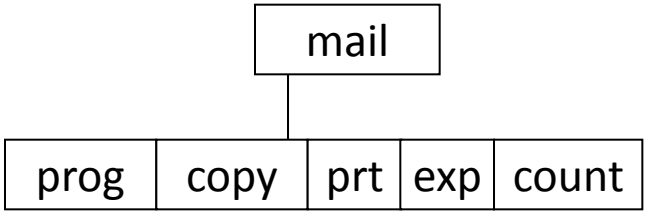
rm <file-name>

Creating a new subdirectory is done in current directory

mkdir <dir-name>

Example: if in current directory /mail

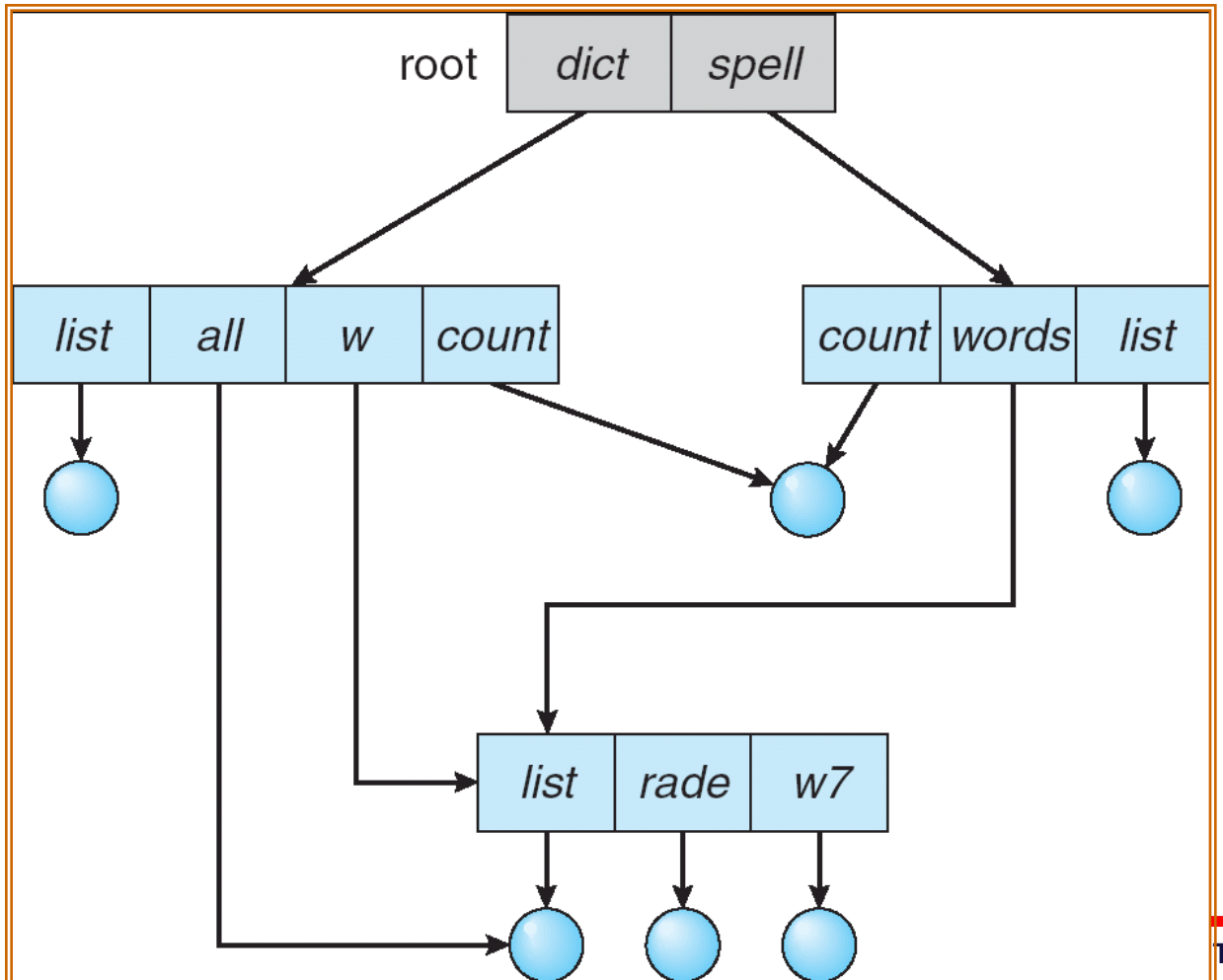
mkdir count



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

Have shared subdirectories and files



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

File Sharing – Multiple Users

User IDs identify users, allowing permissions and protections to be per-user

Group IDs allow users to be in groups, permitting group access rights

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

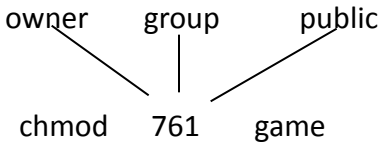
Mode of access: read, write, execute

Three classes of users

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	RWX 1 1 0
c) **public access**	1	⇒	RWX 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.



Attach a group to a file

chgrp G game