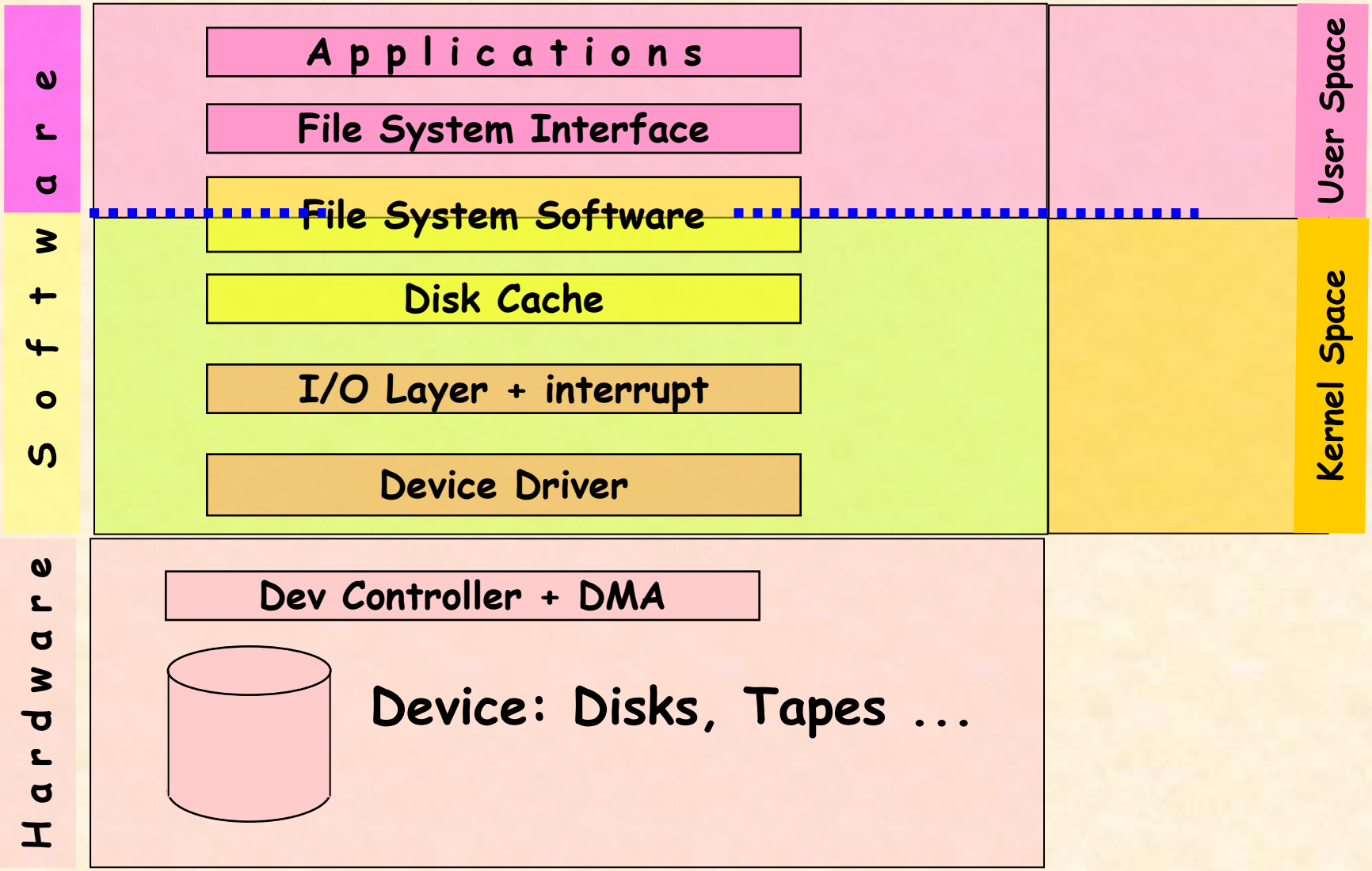
FILE SYSTEMS

File Systems typical layers



File Concept

- Contiguous logical address space
- Files are mapped, by the operating system, onto physical devices
  - These devices are nonvolatile – contents are persistent through power failures and system reboots
- File is a named collection of related information that is recorded on secondary storage
- Types of information stored in a file :
  - Data
    - numeric
    - character
    - binary
  - Programs
    - source programs
    - object programs
    - executable programs

File Structure

- **Text file :**
  - **Is a sequence of characters organized into lines (possibly pages)**
- **Source file :**
  - **Is a sequence of subroutines and functions, which is further organized as declarations followed by executable statements**
- **Object file :**
  - **Is a sequence of bytes organized into 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** – Creation, modification, last use, data for protection, security, and usage monitoring
- Information about files is kept in the directory structure, which is maintained on the disk

File Operations

- File is an abstract data type
- File Operations
  - **Create** : Allocate space, and create directory entry
  - **Write** : Find file location and write from buffer at write pointer
  - **Read** : Find location and read to buffer from read pointer
  - **Reposition within a file** : Move pointer (Seek)
  - **Delete** : Free space, and free directory entry
  - **Truncating a file** : Erase the contents of a file keeping the attributes
  - **Open (FN)** – search the directory structure on disk for entry FN , and move the content of entry to memory.
  - **Close (FN)** – move the content of entry FN in memory to directory structure on disk

Additional Operations

- The directory itself is maintained on disk (normally as a file)
- To avoid the need to search directory for each operation an **open-file table** is maintained
- Open operation creates an entry in this table and returns pointer to this table
- All subsequent operations receive this pointer (instead of name) as parameter
- Close operation releases the entry
- Open file table is normally split in two levels of internal tables
  - **System-wide table** (**GOFT** - global open file table)
  - **Per-Process table** (**OFT** open file table)

Open Files

- **System-wide table** – contains process independent information such as location of the file on disk, file size, aceess dates
- **Per-Process table** – Tracks all files that a process has open. It has information regarding the use of the file by the process such as current file pointer, access rights

✓ **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**: 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

Open File Locking

- Facility provided by some operating systems and file systems for locking an open file (or part of a file)
- Prevents other process from gaining access to a open file
- File locking may be either **Mandatory** or **Advisory**:
  - **Mandatory** – access is denied depending on locks held and requested. Operating system ensures locking integrity Ex. Windows Systems
  - **Advisory** – processes can find status of locks and decide what to do. It is upto the software developer to ensure locking integrity. Ex. UNIX systems

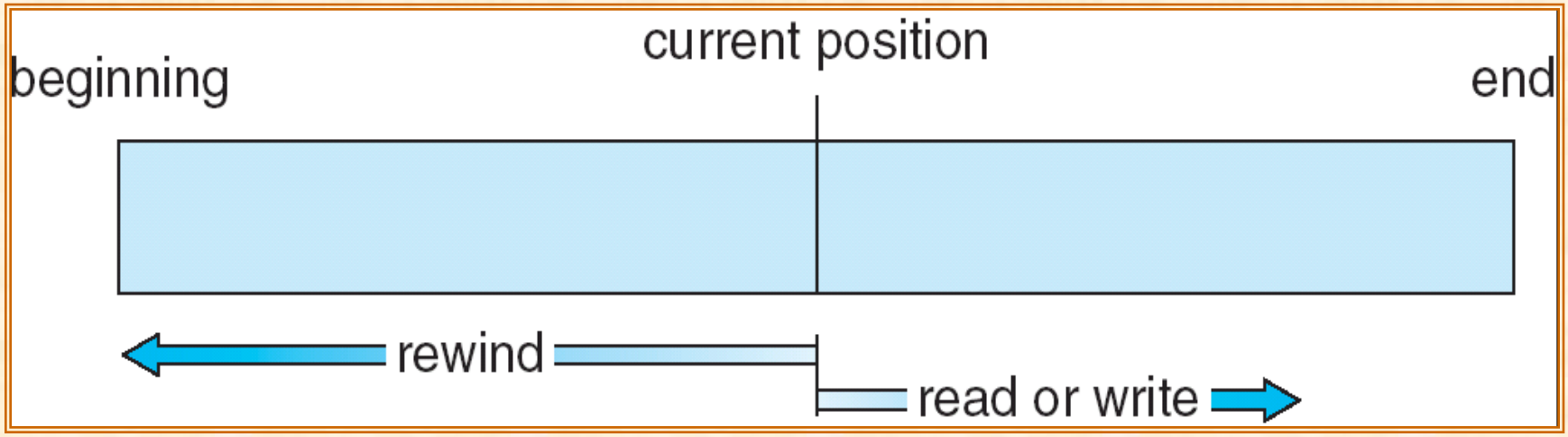
File Types – Name, Extension

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

Access Methods

1. **Sequential Access :**
   - **simplest access method**
   - **Information in the file is processed in order, one record after the other**
   - **Example : editors and compilers access files sequentially**

✓ **Operations on files are :** **read next**
write next
reset
no read after last write (rewrite)



Access Methods

2. **Direct Access :**
   - **Also called relative access**
   - **Read or write takes in no particular order**
   - **File is viewed as numbered sequence of blocks or records**
   - **Example : databases**

✓ **Operations on files are :**

read *n*

write *n*

position to *n*

read next

write next

rewrite *n*

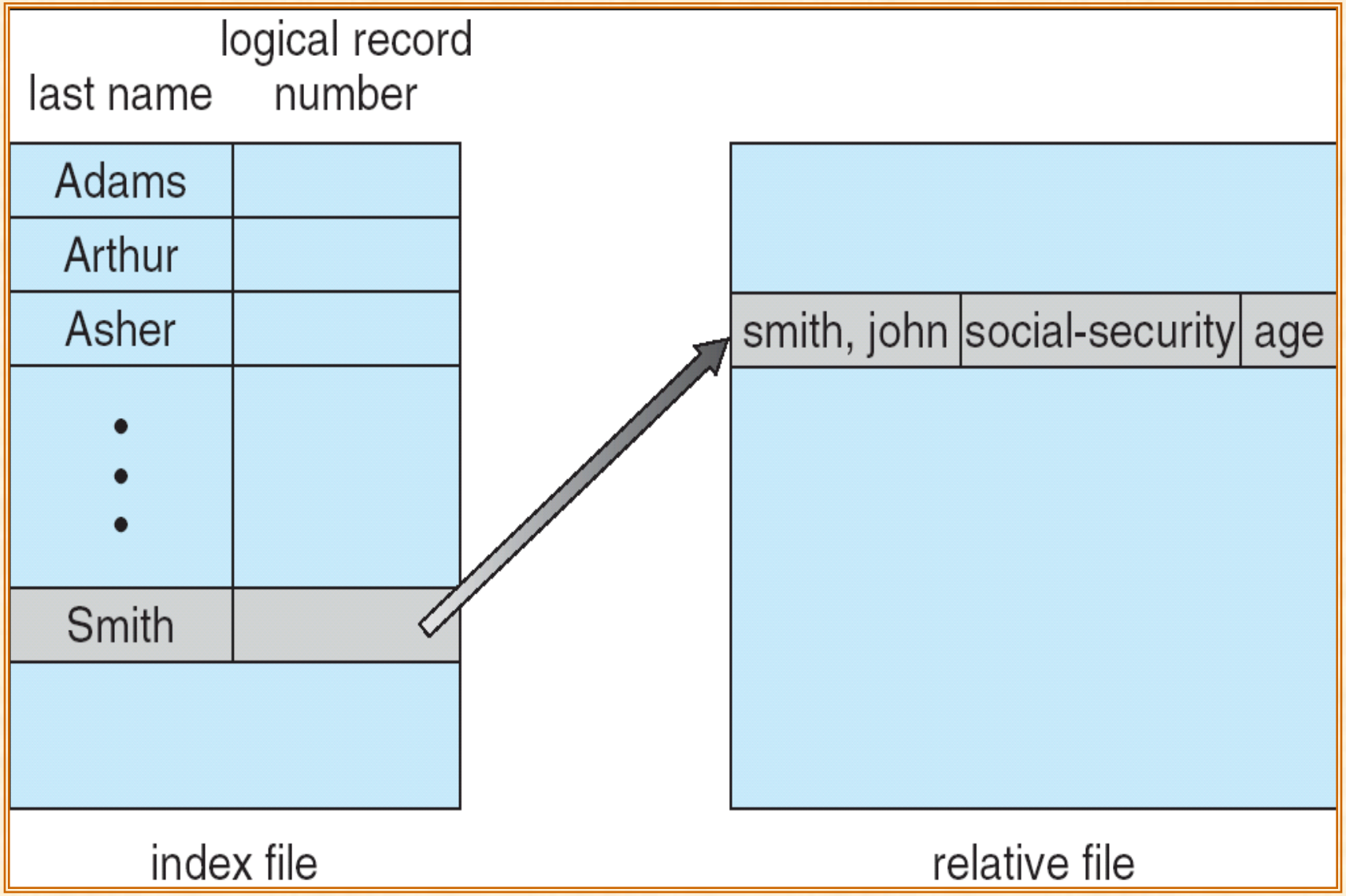
***n* = relative block number**

Simulation of Sequential Access on a Direct-access File

- ✓ **Cp defines our current position**
- ✓ **We can simulate sequential file operations as follows:**

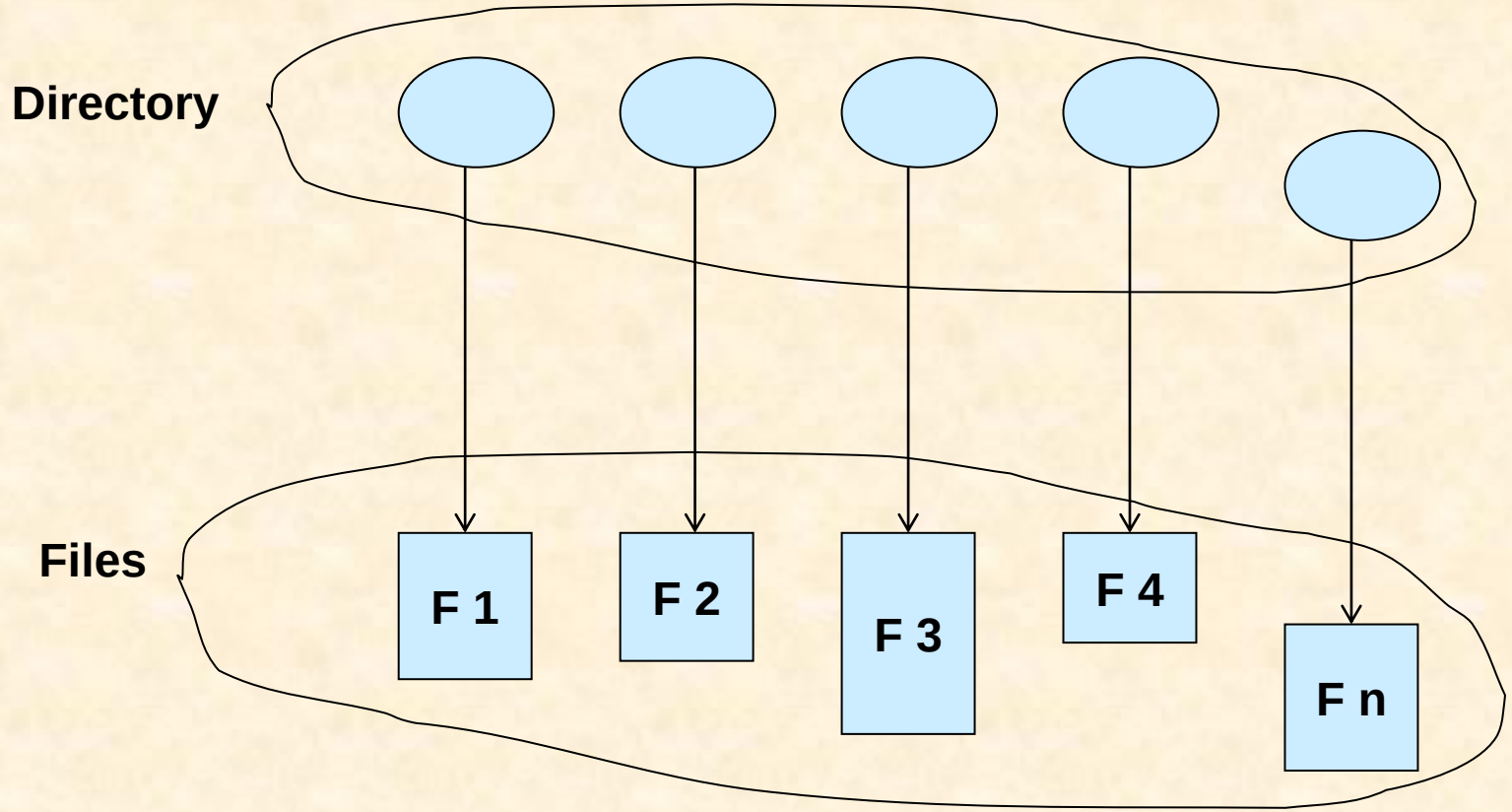
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Example of Index and Relative Files

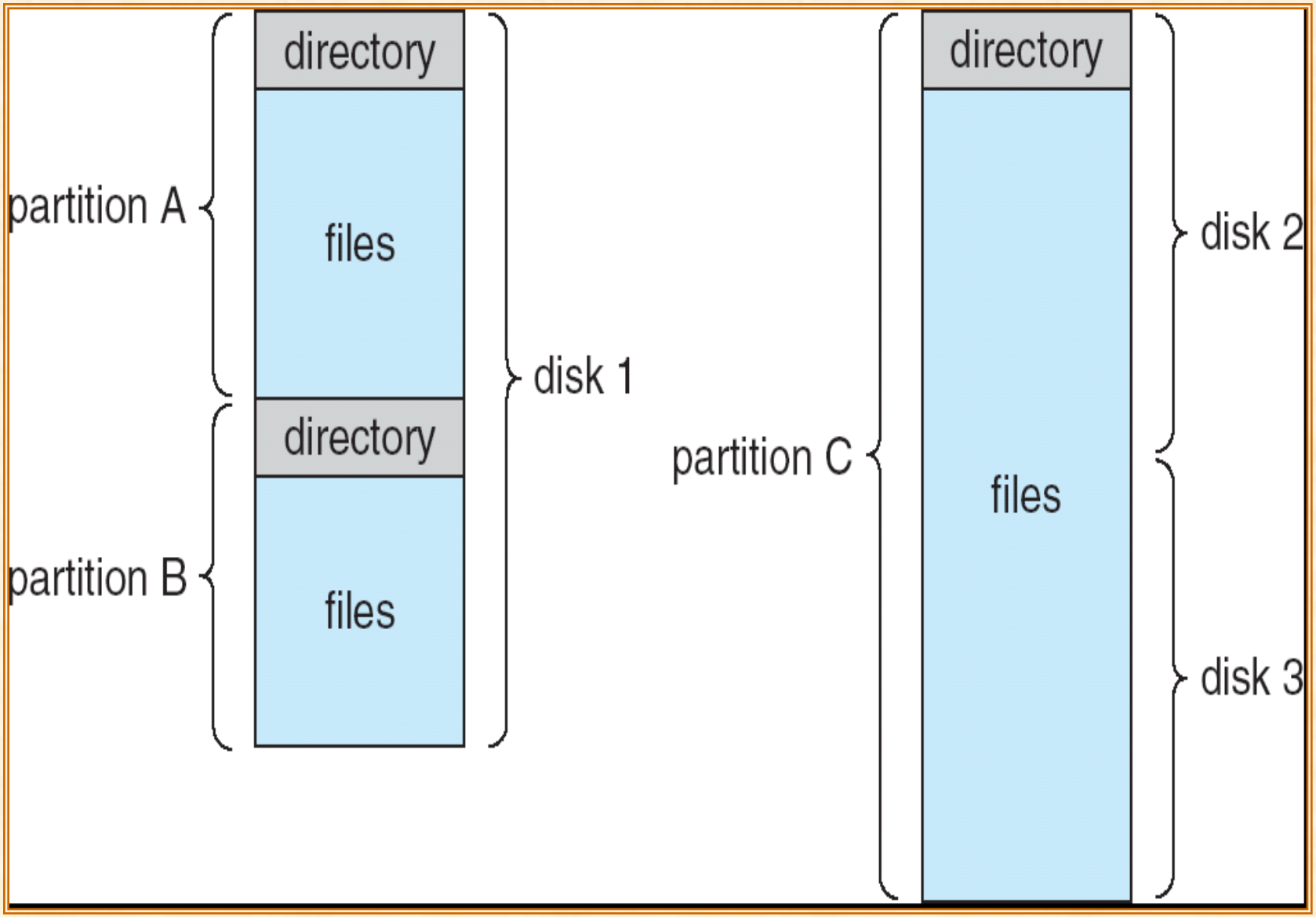


Directory Structure

- ✓ A collection of nodes containing information about all files
- ✓ Both the directory structure and the files reside on disk



A Typical File-system Organization



Operations Performed on Directory

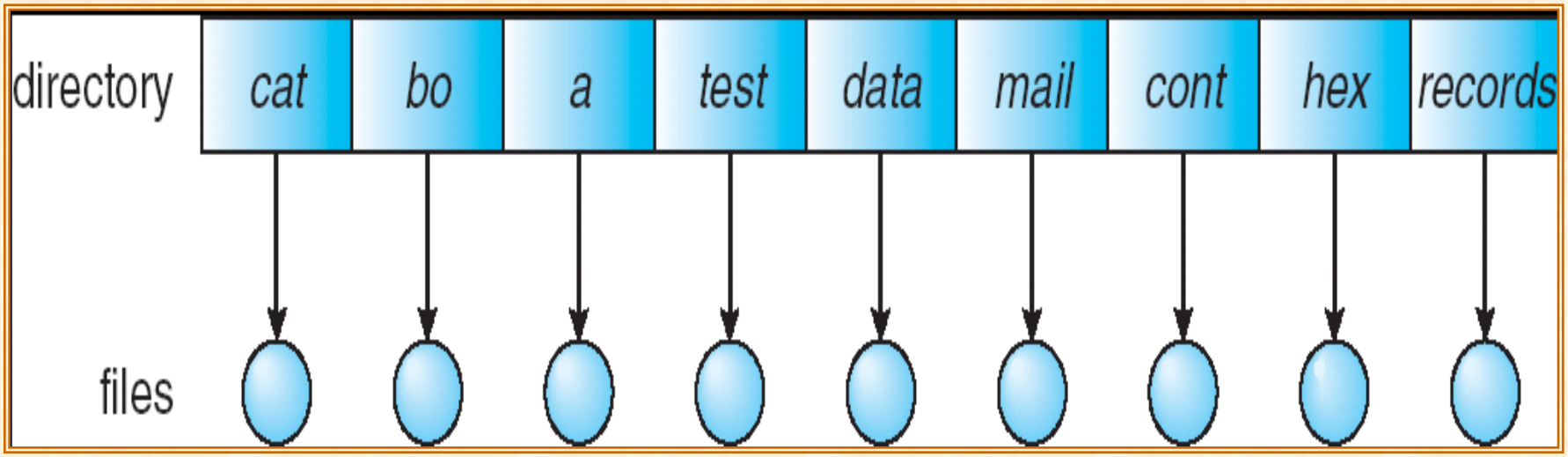
- ✓ **Search for a file**
- ✓ **Create a file**
- ✓ **Delete a file**
- ✓ **List a directory**
- ✓ **Rename a file**
- ✓ **Traverse the file system**

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

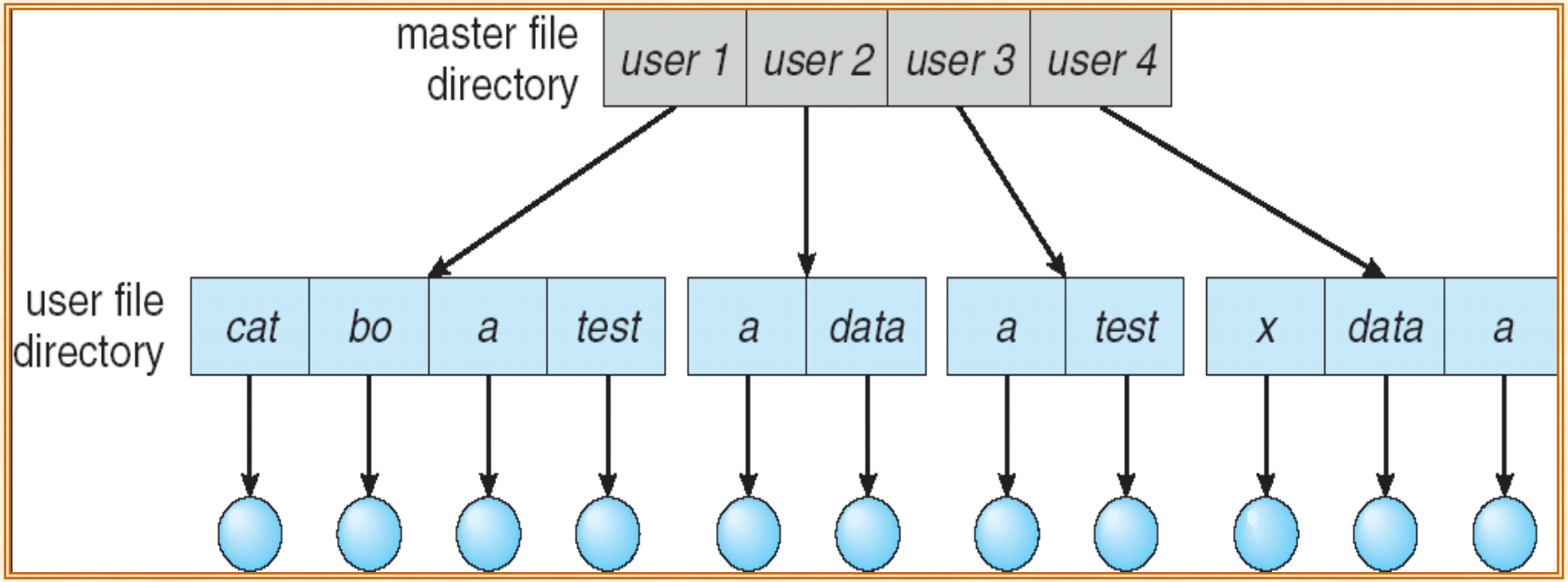
- ✓ **Simplest directory structure**
- ✓ **A single directory for all users**
- ✓ **All files are contained in the same directory**



- ✓ **Naming problem**
- ✓ **Grouping problem**

Two-Level Directory

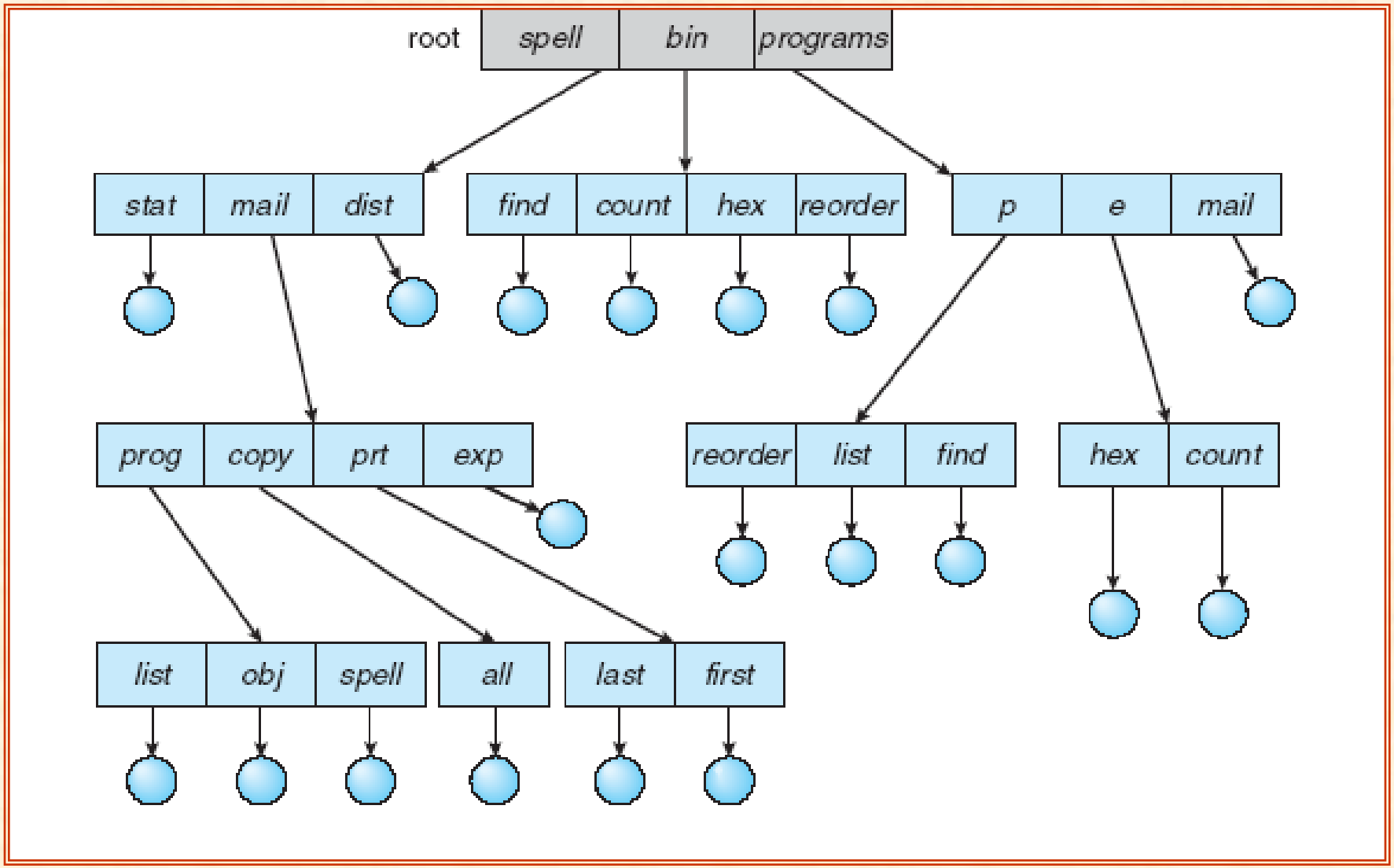
- ✓ **Separate directory for each user**



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

Tree-Structured Directories

Most common directory structure



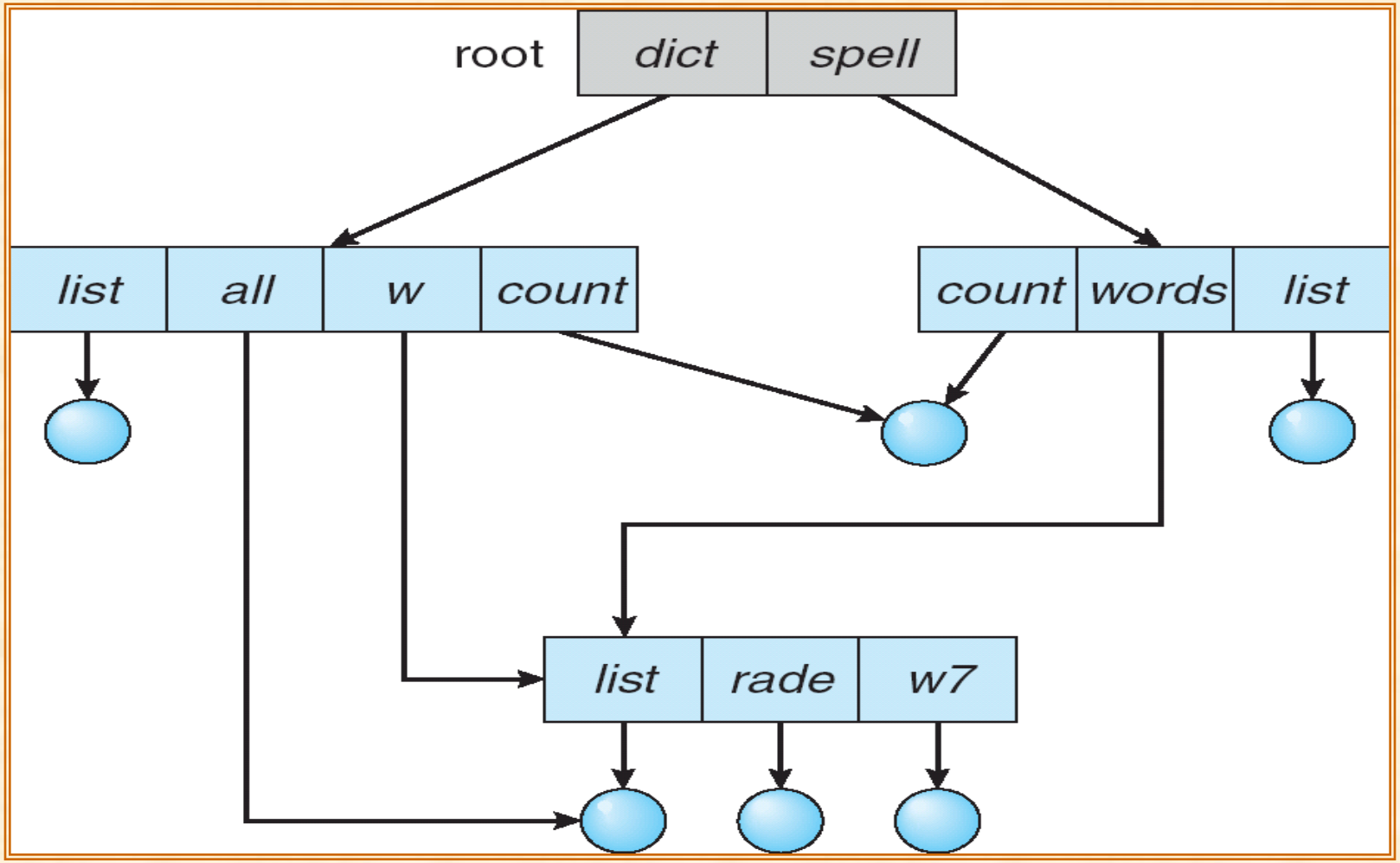
Path names

- Path names can be of 2 types:
  - Absolute
    - Begins at the root and follows a path down to the specified file
  - Relative
    - Defines path from current directory

- Advantages of Tree-Structured Directories
  - Efficient searching
  - Grouping Capabilities

Acyclic-Graph Directories

- ✓ **Allows directories to share subdirectories and files**



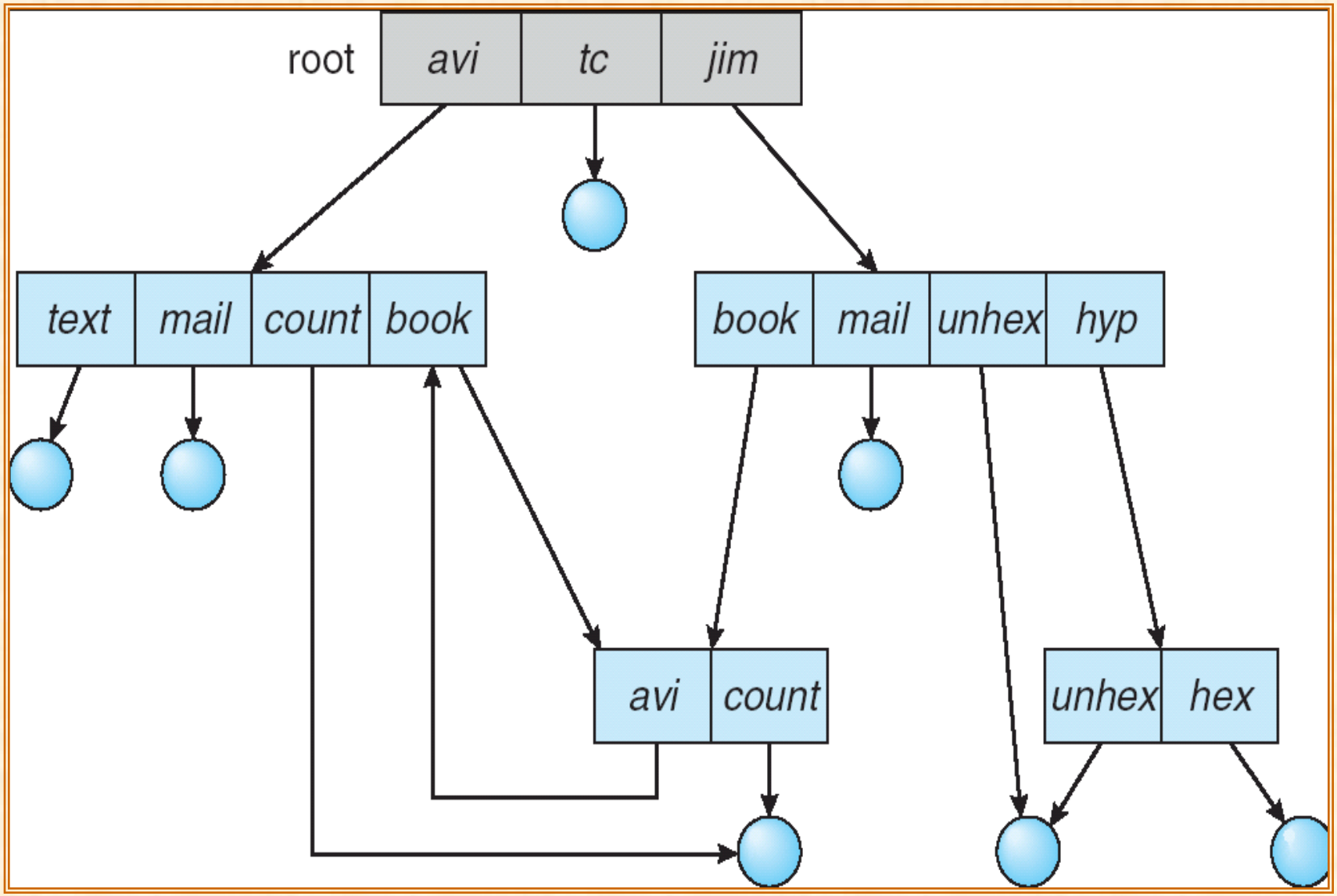
Acyclic-Graph Directories (Cont.)

- Tree structure prohibits the sharing of files or directories
  - Acyclic graph allows directories to have shared subdirectories and files
- Shared files and subdirectories can be implemented in several ways: a common way is :
  - Create a new directory called a *link*
    - Link is effectively a pointer to another file or sub directory
    - Link may be implemented as absolute or relative path name (a symbolic link)
  - Duplicate all the information about them in both sharing directories
    - A link is clearly different from the original directory entry; thus 2 are not equal
    - It makes original and the copy indistinguishable
    - Problem:
      - Maintaining consistency if the files is modified

Acyclic-Graph Directories (Cont.)

- Disadvantages:
  - File may now have multiple absolute path names, distinct file names may refer to the same file
    - Trying to traverse the entire file system becomes significant, since we do not want to traverse shared structures more than once
  - Deletion
    - When can the space allocated to a shared file be deallocated and reused ?
    - Remove the file whenever anyone deletes it, this may leave dangling pointers to the now-nonexistent file
    - Different approaches:
      - search for the links and remove them
      - preserve the file until all references to it are deleted

General Graph Directory



General Graph Directory (Cont.)

- Problem with using acyclic graph structure is ensuring that there are no cycles
- 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

Protection

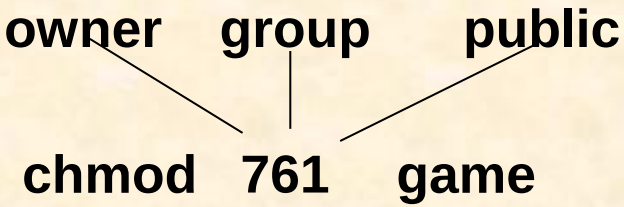
- When information is kept in computer system , a major concern is its:
  - Protection from both physical damage (reliability) and
  - Improper access (protection)
  - Reliability is provided by duplicate copies of files
  - Protection can be provided in many ways
    - For a small single user system we might provide protection by physically removing the floppy disks and locking them in a desk drawer or file cabinet
- 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**
c) public access	**1**	**⇒**	**0 0 1**

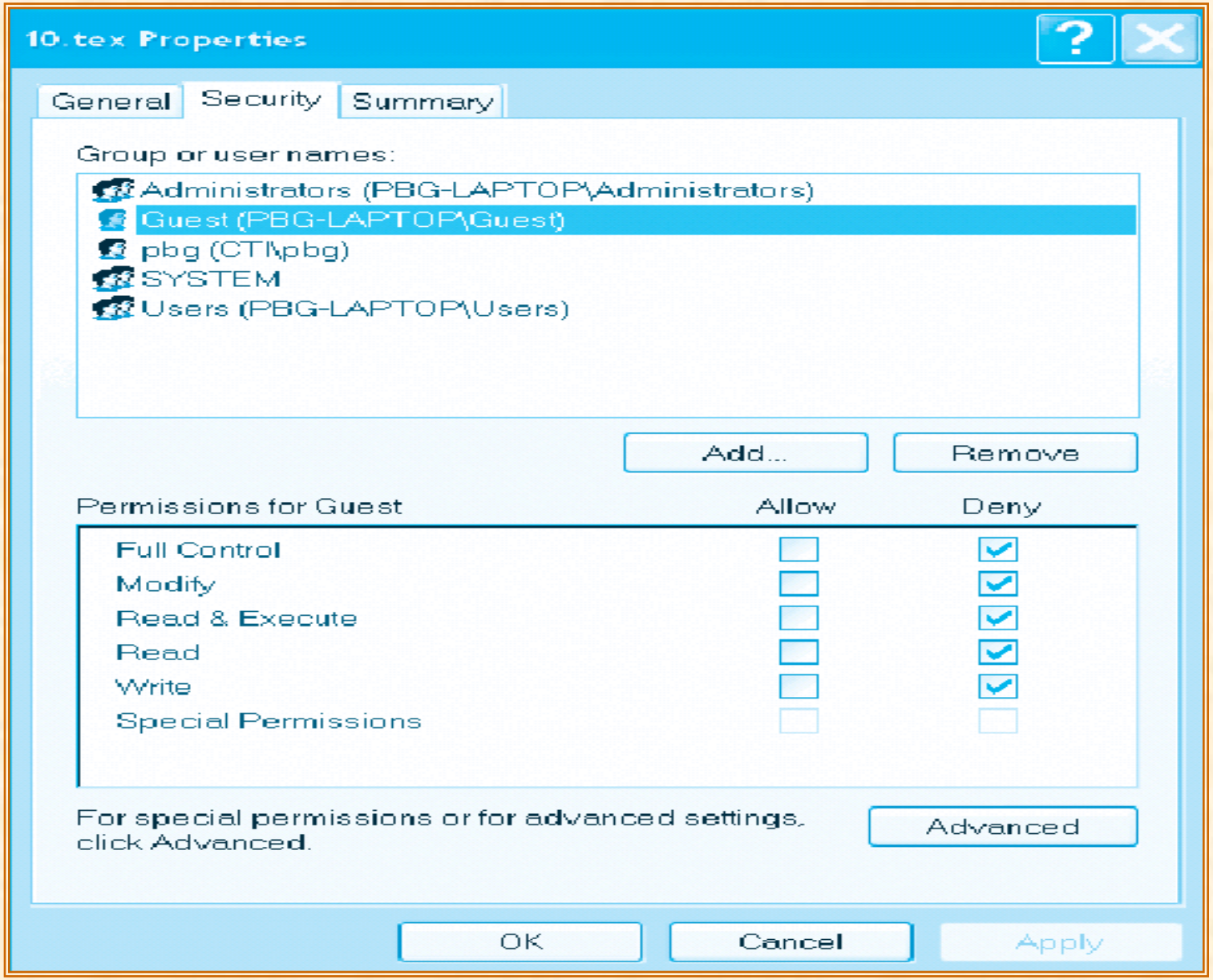
- ✓ **Ask manager to create a group (unique name), say G, and add some users to the group.**
- ✓ **For a particular file (say *game*) or subdirectory, define an appropriate access.**



Attach a group to a file

chgrp G game

Windows XP Access-control List Management



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Consistency Semantics

- Defines what happens when number of users try to access file simultaneously
- **Session** - set of reads and writes within one open and close file
- **UNIX**
  - A write to a file is immediately visible to other users
  - A number of users may share same read/write pointer
  - Implemented by a single image for each file
- **AFS** - Andrew File System (distributed)
  - A write is not seen immediately by other users
  - When file is closed, updates visible for sessions which start after file was closed
  - There are number of file images

Consistency Semantics

- **Immutable shared files semantics:**
  - **Once a file is declared as shared by its creator , it cannot be modified**

- **2 important properties:**
  - **Its name may not be reused and**
  - **Its contents may not be altered**

End of the Chapter