PART FOUR
STORAGE MANAGEMENT

STORAGE MANAGEMENT

- □ Chapter 10 File-System Interface
- □ Chapter 11 File-System Implementation
- □ Chapter 12 Mass-Storage Structure
- □ Chapter 13 I/O Systems

Chapter 10
File-System Interface

Contents

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

Objectives

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore file-system protection

File Concept

- □ Information is important.
- □ A file is a named collection of related information that is recorded on secondary storage.
- □ It is a contiguous logical address space
- □ From a user’s perspective, a file is the smallest allotment of logical secondary storage.

File system

- Services that File System provided for users
  - 文件访问：文件的创建、打开和关闭，文件的读写；
  - 目录管理：用于文件访问和控制的信息，不包括文件内容
  - 文件结构管理：划分记录，顺序，索引
  - 访问控制：并发访问和用户权限
  - 限额(quota)：限制每个用户能够建立的文件数目、占用外存空间大小等
  - 审计(auditing)：记录对指定文件的使用信息（如访问时间和用户等），保存在日志中

File system

- Modules in File System
  - 文件的分块存储：与外存的存储块相配合
  - I/O缓冲和调度：性能优化
  - 文件定位：在外存上查找文件的各个存储块
  - 外存存储空间管理：如分配和释放。主要针对可改写的外存如磁盘。
  - 外存设备访问和控制：包括由设备驱动程序支持的各种基本文件系统如硬盘，软盘，CD ROM等

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
- □ Typically, a directory entry consists of the file’s name and its unique identifier. The identifier in turn locates the other file attributes.

File Operations

- File is an **abstract data type**
- **Basic operations (the minimal set of required file operations)**
  - **Create**
  - **Write**
  - **Read**
  - **Reposition within file**
  - **Delete**
  - **Truncate**
- **Other operations**
  - **Appending**
  - **Renaming**
  - **Copy**
  - **Open**
  - **Close**
- *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 file table

Open Files

- 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
- File locks allow one process to lock a file and prevent other processes from gaining access to it.
- File locks provided functionality similar to reader-writer locks.
- A shared lock is akin to a reader lock in that several processes can acquire the lock concurrently.
- An Exclusive lock behaves like a writer lock.
- Mediates access to a file
- Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested
  - **Advisory** – processes can find status of locks and decide what to do

File Locking Example – Java API

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
                RandomAccessFile raf = new RandomAccessFile("file.txt",
    "rw");
                // get the channel for the file
                FileChannel ch = raf.getChannel();
                // this locks the first half of the file - exclusive
                exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
                /** Now modify the data . . . */
                // release the lock
                exclusiveLock.release();
```

File Locking Example – Java API (cont)

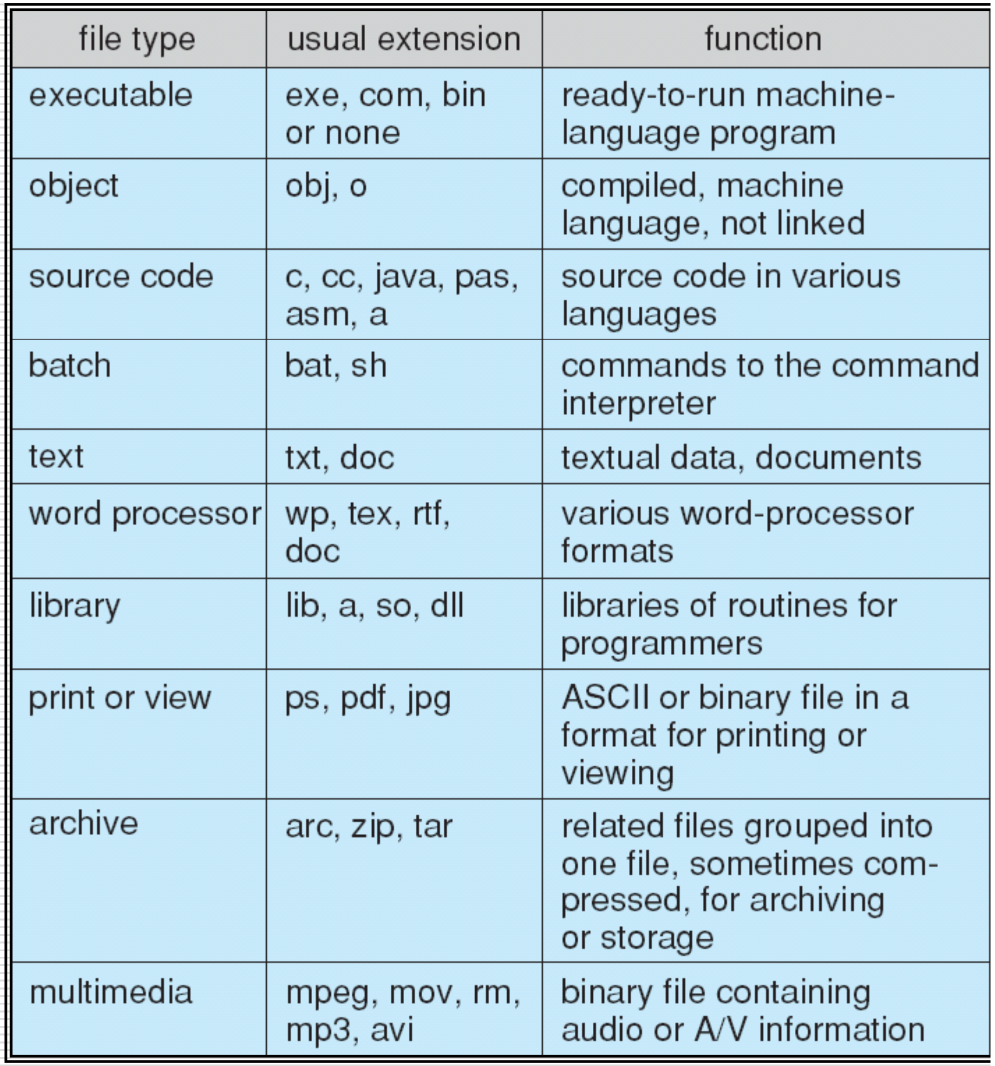
```
                    // this locks the second half of the file - shared
                    sharedLock = ch.lock(raf.length()/2+1, raf.length(),
                            SHARED);
                    /** Now read the data . . . */
                    // release the lock
                    sharedLock.release();
            } catch (java.io.IOException ioe) {
                    System.err.println(ioe);
            }finally {
                    if (exclusiveLock != null)
                    exclusiveLock.release();
                    if (sharedLock != null)
                    sharedLock.release();
            }
        }
}
```

File Types

- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

File Types – Name, Extension

- Only if the OS recognizes the type of a file, it can then operate on the file in reasonable ways.
- A common technique for implementing file types is to include the type as part of the file name.



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 Types

- In Mac OS X, each file has a creator attribute containing the name of the program that created it.
- The UNIX system uses a crude magic number stored at the beginning of some files to indicate roughly the type of the file.

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

File Structure

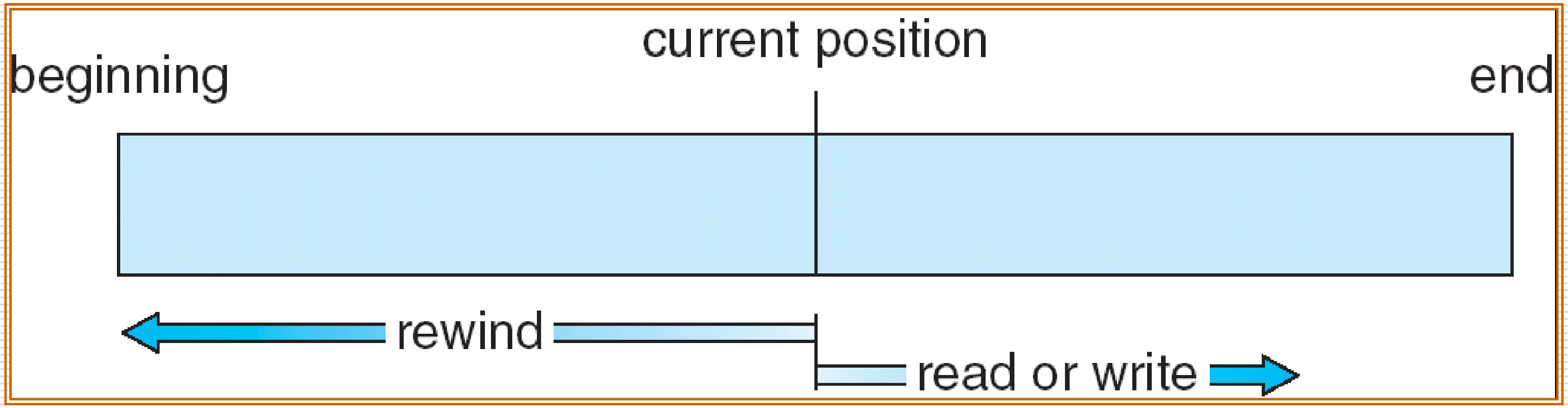
- A text file is a sequence of characters organized into lines and possibly pages.
- A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements.
- An object file is a sequence of bytes organized into blocks understandable by the system’s linker.
- An executable file is a series of code sections that the loader can bring into memory and execute.

Access Methods

- □ **Sequential Access**
  - read next
  - write next
  - reset
  - no read after last write
    - (rewrite)
- □ **Direct Access**
  - read *n*
  - write *n*
  - position to *n*
    - read next
    - write next
  - rewrite *n*

n = relative block number

Sequential-access File



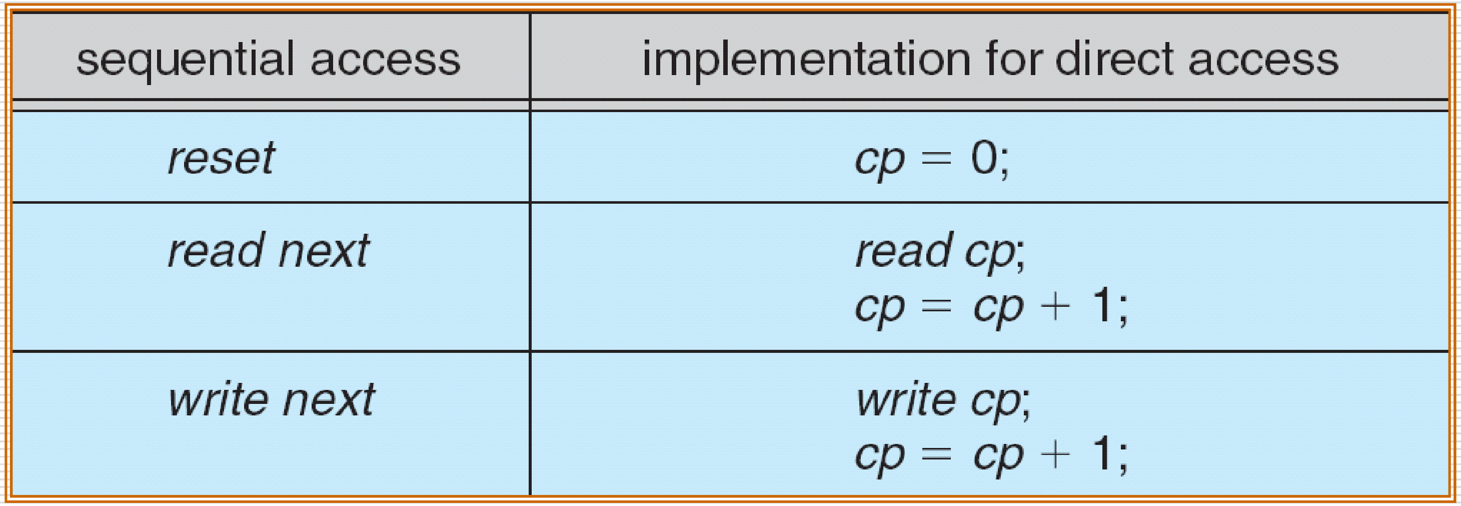
Simulation of Sequential Access on a Direct-access File

Read n

Write n

Position to n

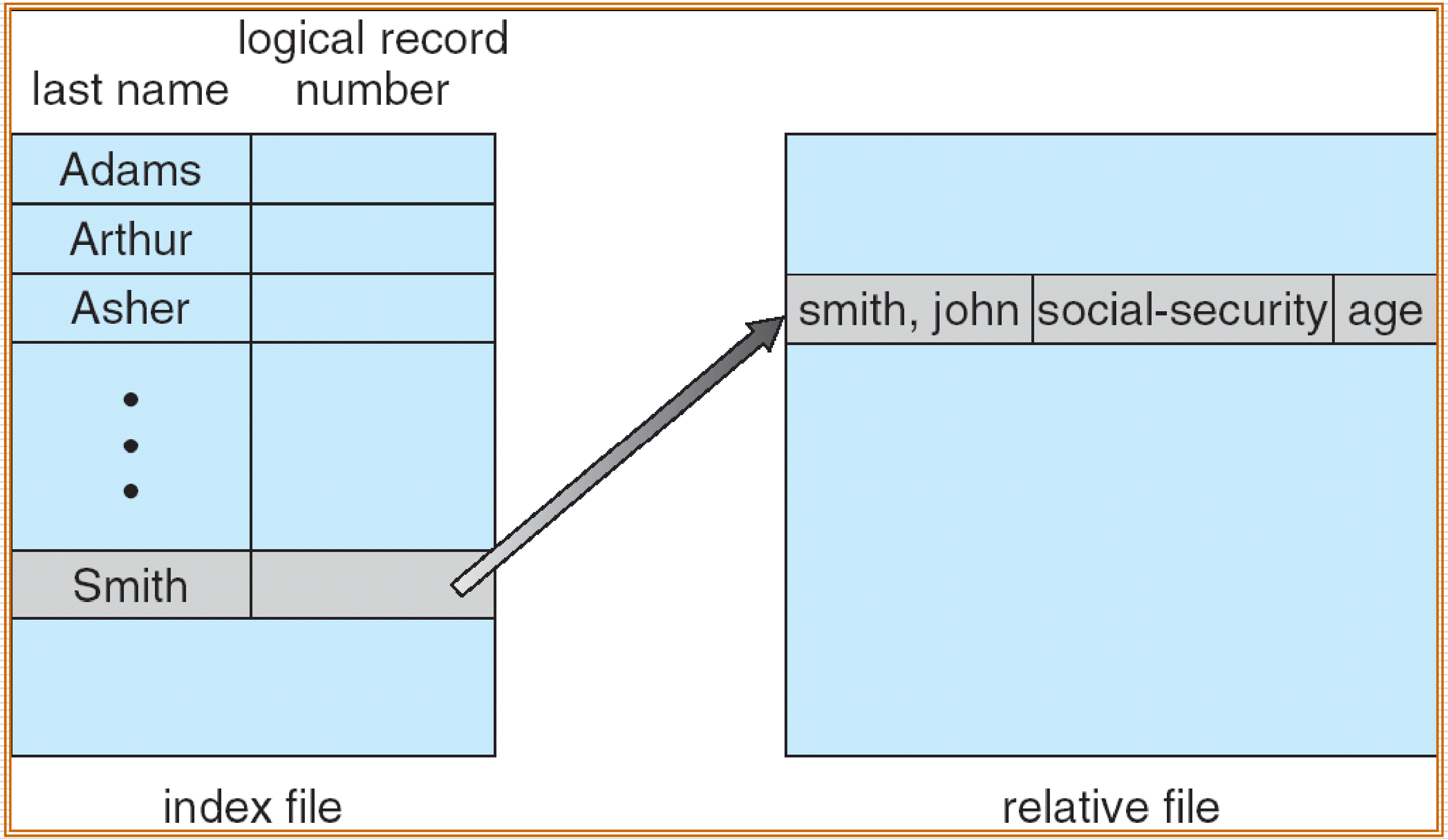
- Read next
- Write next



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

- Index: like an index in the back of a book. To find a record in the file, we first search the index, and then use the pointer to access the file directly and to find the desired record

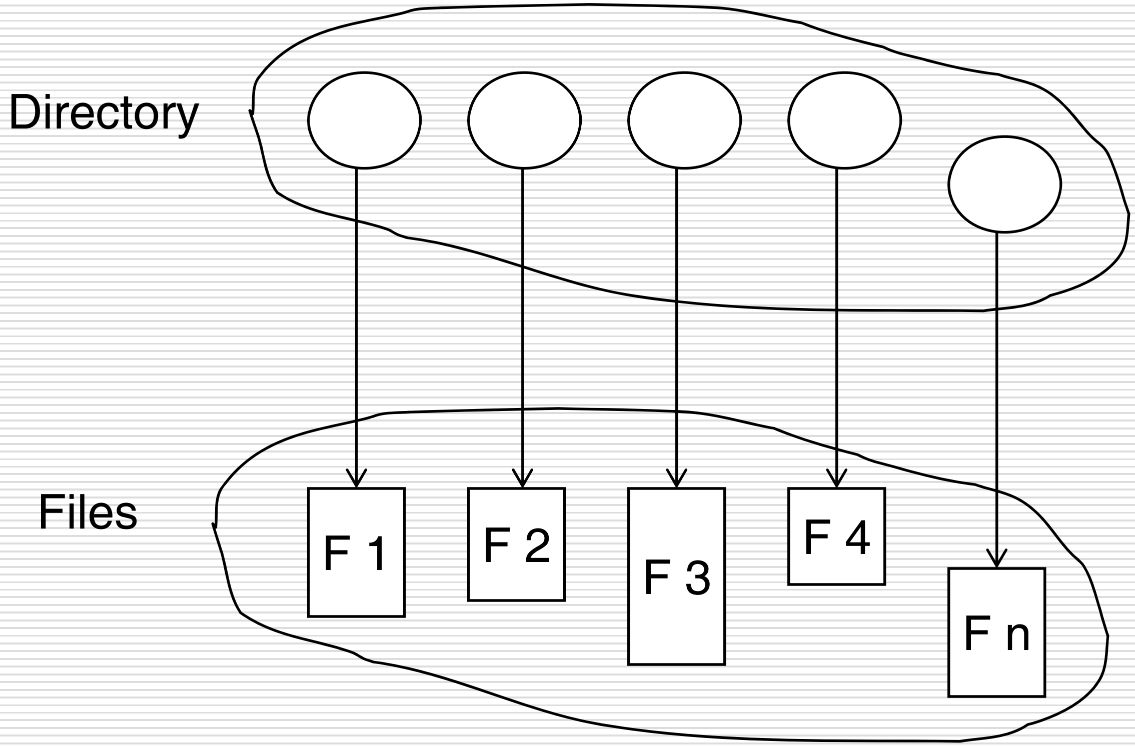


Other methods

- With large files, the index file itself may become too large to be kept in memory.
  - One solution: to create an index for the index file.
  - For example, IBM’s indexed sequential-access method(ISAM) uses a small master index that points to disk blocks of a secondary index

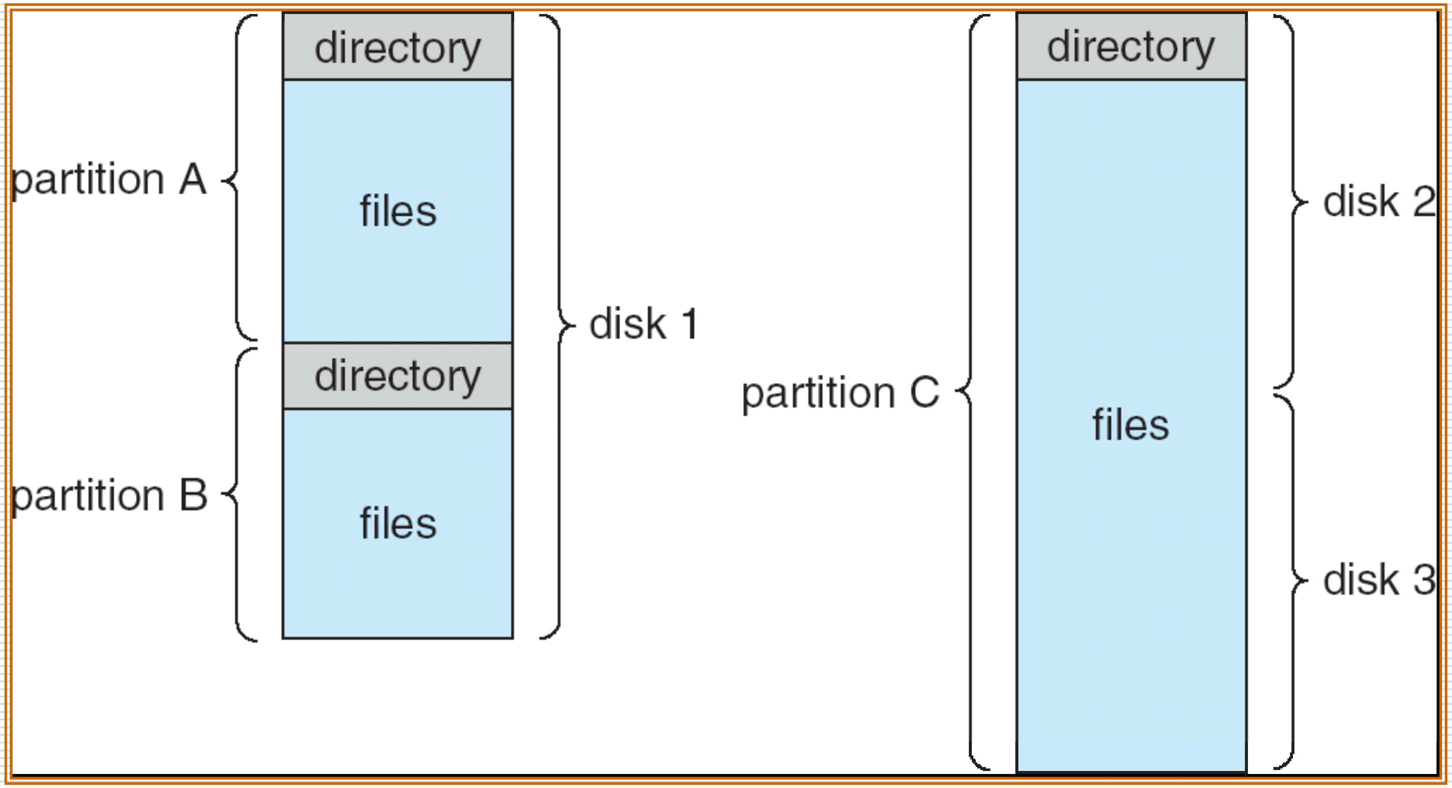
10.3 Directory Structure

- A collection of nodes containing information about all files



Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

A Typical File-system Organization



Information that directory usually contains

- Name
- Type
- Address
- Current Length
- Maximum Length
- Access Date
- Update Date
- User Identifier
- Protection

Operations Performed on Directory

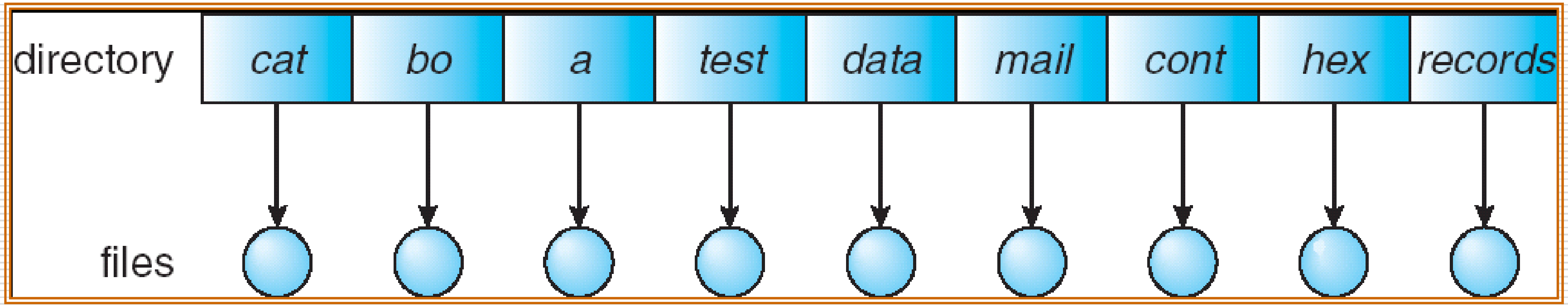
- □ Search for a file
- □ Create a file
- □ Delete a file
- □ List a directory
- □ Rename a file
- □ Traverse the file system

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

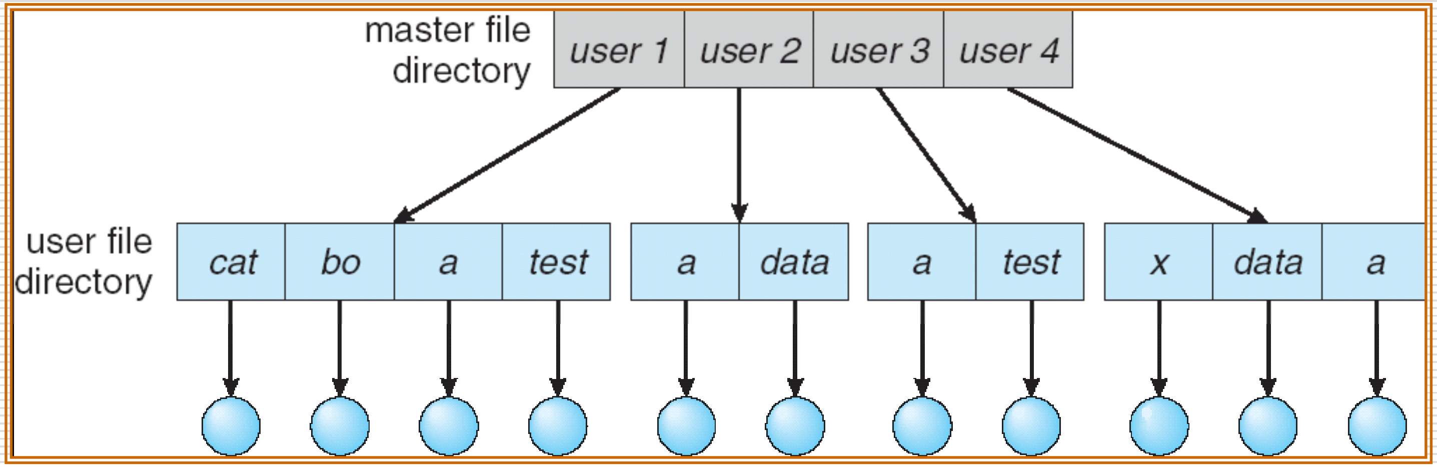
- □ A single directory for all users



- □ Naming problem
- □ Long time to search long directory

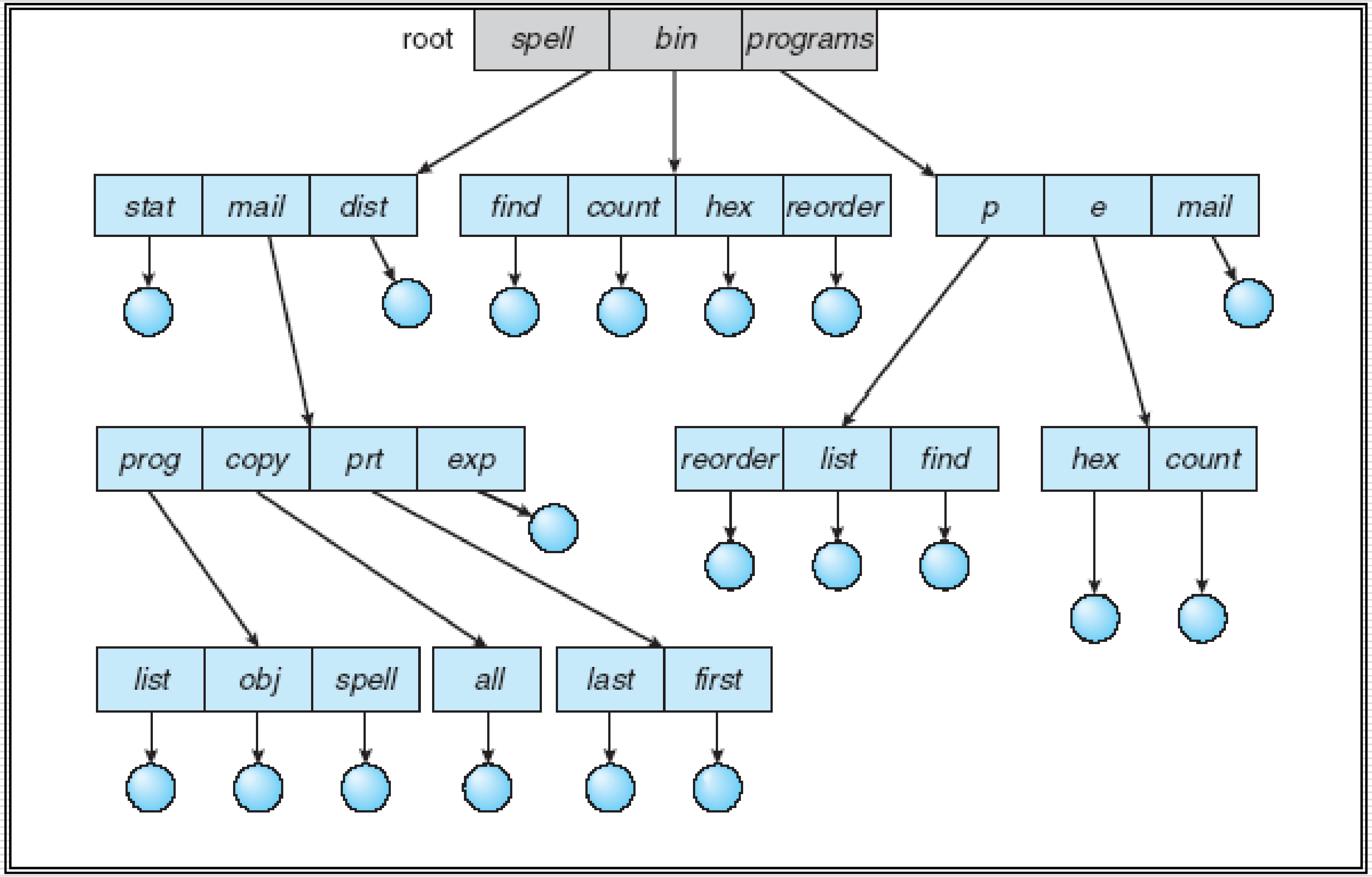
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- Cannot group files

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Efficient searching

- Grouping Capability

- Current directory (working directory)
  - cd /spell/mail/prog
  - type list
- parent directory (父目录)、
- subdirectory (子目录)、
- root directory (根目录)

Tree-Structured Directories (Cont)

- □ **Absolute** or **relative** path name
- □ Creating a new file is done in current directory
- □ Delete a file

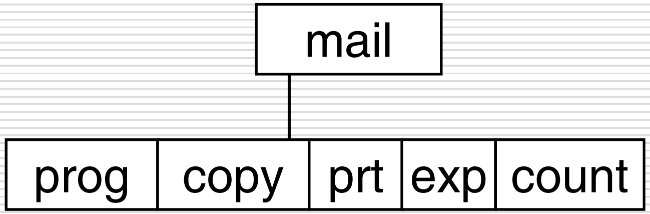
  rm <file-name>

- □ Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

  mkdir count



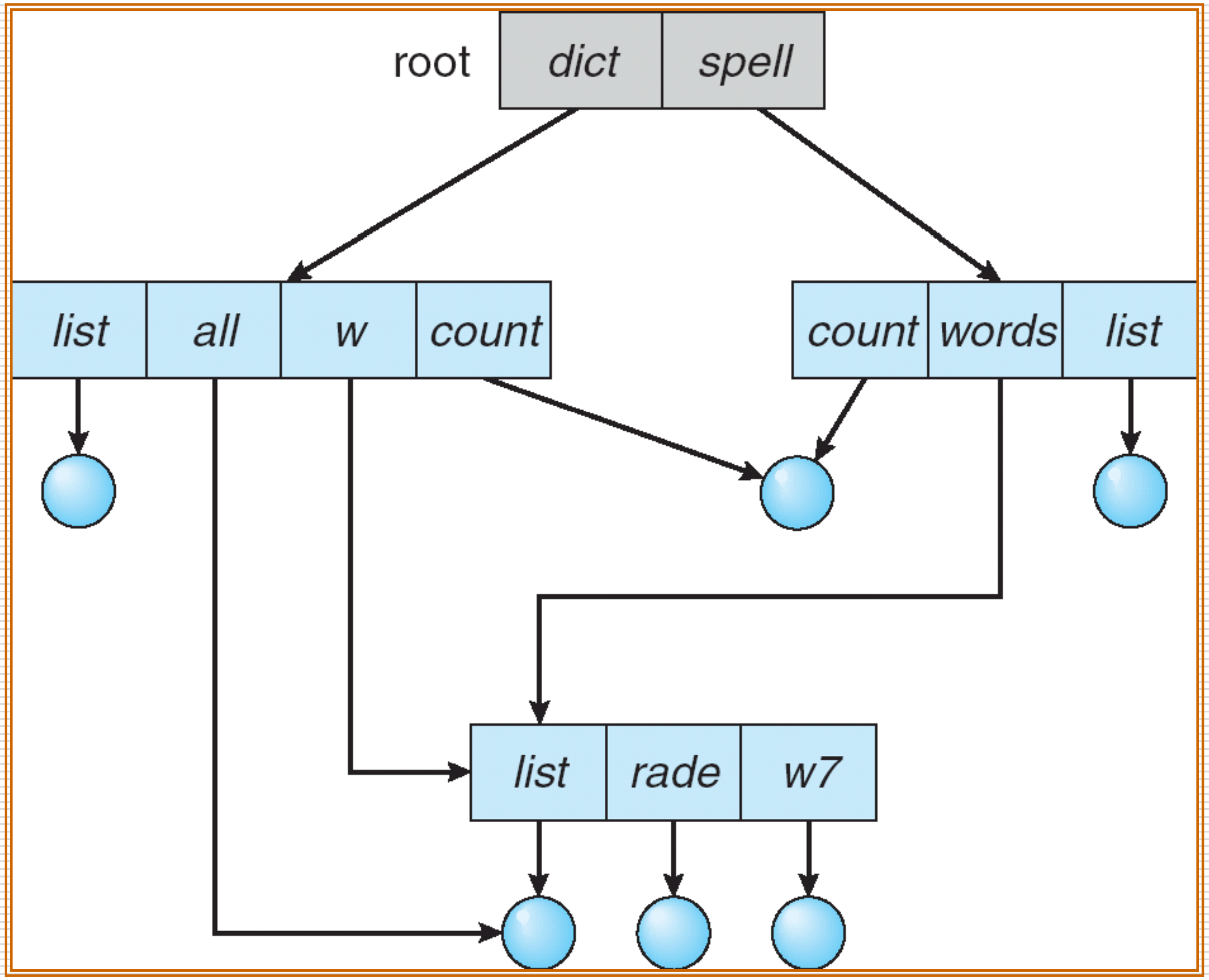
Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Tree-Structured Directories (Cont)

- How to delete a directory, such as /mail?
- If a directory is empty, its entry in its containing directory can simply be deleted
- If a directory is not empty, there are two approaches:
  - Can not delete a directory unless it's empty. Such as MS-DOS
  - Provide an option that all directory's files and subdirectories are also to be deleted. Such as UNIX rm command
    - It's more convenient, but more dangerous

Acyclic-Graph Directories

- Have shared subdirectories and files



Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)

- If *dict* deletes *list* ⇒ dangling pointer

  Solutions:

  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Entry-hold-count solution

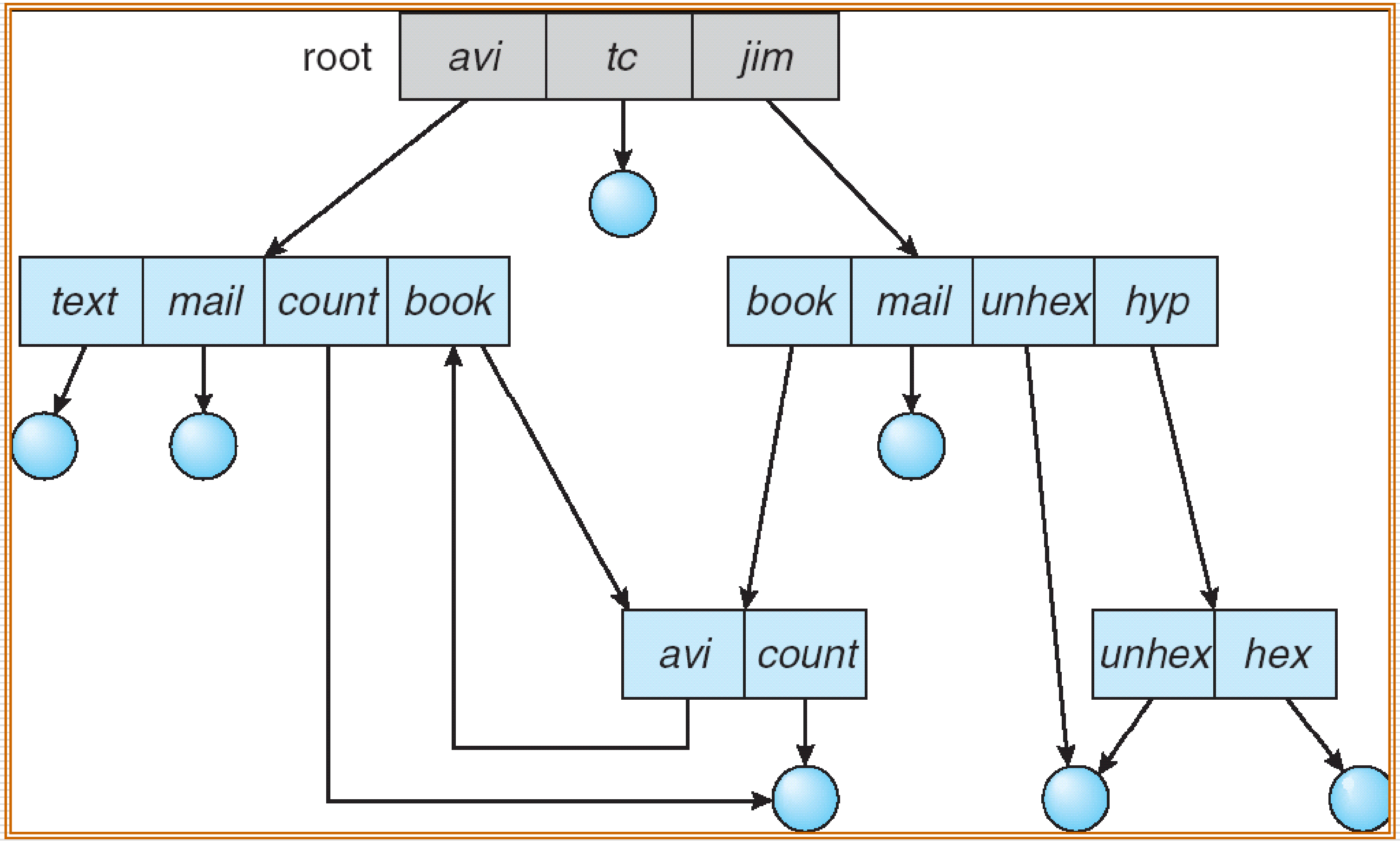
Shared files or directories

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

Other problems should be considered

- □ One file maybe have several names
- □ When can the space allocated to shared files be reused?
  - ■ We can maintain a counter.
  - ■ We can search the list

General Graph Directory



General Graph Directory (Cont.)

- Problems
  - How to determine whether a file can be deleted.
- solution
  - Garbage collection
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

Windows FAT32目录结构

- Windows98、2000长文件名的目录项由几个MS－DOS的32B目录项组成。
- 它用一个表项存放短文件名和这个文件或子目录的其它信息（包括MS－DOS FCB原有的簇号、文件大小，最后修改时间和最后修改日期，还有Windows98增加的创建时间、创建日期和最后存取日期），短文件名的属性是0x20。
- 用连续若干个表项存放长文件名，每个表项存放13个字符（使用Unicode编码，不论西文和汉字，每个字符一律占用2个字节。对西文第一个字节存ASCII码，第二个字节存0x00。）
- 长文件名的表项首字节的二进制数低5位值，分别为00001B 、00010B 、 00011B 、……，表示它们的次序，左起第2位为1（也就是在低5位基础上加40H）表示该表项是最后一项。最后项存放13个字符位置多余时，先用文件名项的第13、272个字节0表示结束，再用FFH填充。长文件名的属性是0FH。长、28字节为0x00，第14字节为短文件名检验和。
- 长文件名The quick brown.fox（短文件名为THEQUI~1.FOX）目录项格式如下：

Windows FAT32目录结构

<table>
<tr><td>42</td><td colspan="2">w
77 00</td><td colspan="2">n
6E 00</td><td colspan="2">2E 00</td><td colspan="2">f
66 00</td><td colspan="2">o
6F 00</td><td>属性
0F</td><td>00</td><td>检验和
07</td><td colspan="2">X
78 00</td></tr>
<tr><td colspan="2">00 00</td><td colspan="2">FF FF</td><td colspan="2">FF FF</td><td colspan="2">FF FF</td><td colspan="2">FF FF</td><td colspan="2">00 00</td><td colspan="2">FF FF</td><td colspan="2">FF FF</td></tr>
<tr><td>01</td><td colspan="2">T
54 00</td><td colspan="2">h
68 00</td><td colspan="2">e
65 00</td><td colspan="2">20 00</td><td colspan="2">q
71 00</td><td>属性
0F</td><td>00</td><td>检验和
07</td><td colspan="2">U
75 00</td></tr>
<tr><td colspan="2">i
69 00</td><td colspan="2">c
63 00</td><td colspan="2">k
6B 00</td><td colspan="2">00</td><td colspan="2">b
62 00</td><td colspan="2">00 00</td><td colspan="2">r
72 00</td><td colspan="2">0
6F 00</td></tr>
<tr><td colspan="8">短文件名
T H E Q U I ~ 1</td><td colspan="3">扩展名
F 0 X</td><td>属性
20</td><td></td><td colspan="3">创建时间</td></tr>
<tr><td colspan="2">创建日期</td><td colspan="2">最后存取日期</td><td colspan="2">00 00</td><td colspan="2">最后修改时间</td><td colspan="2">最后修改日期</td><td colspan="2">第一簇号</td><td colspan="4">文件大小</td></tr>
</table>

UNIX的树型目录

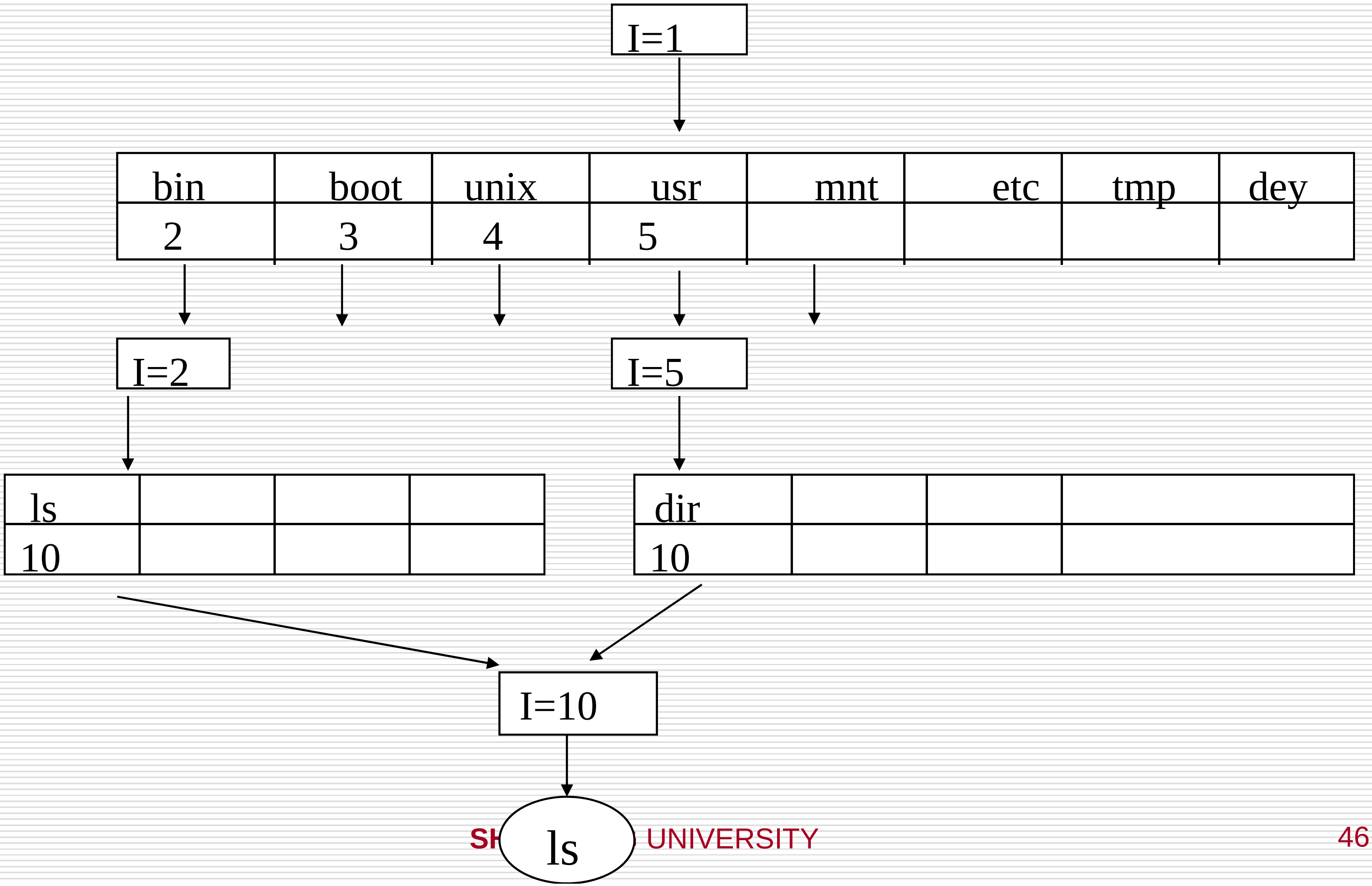
- □ UNIX为了加快目录的寻找速度，将文件控制块FCB中文件名和文件说明分开。文件说明为索引节点，各文件索引节点集中存放在索引节点区，索引节点按索引节点号排序。而文件名与索引节点号构成目录，UNIX S V 操作系统的文件名为14个字节，索引节点2个字节，共16个字节构成目录项。同一级目录构成目录文件，在文件区存放。
- □ Linux目录文件中的目录项会变长，以保证系统支持文件名长度可变，最长达255个字符。目录项的前三项是定长的，包含以下信息：(1)索引节点号(4B)；(2)目录项长度(2B)；(3)文件名长度(2B)。目录项最后是文件名，目录项不能跨越二个块 。

UNIX的树型目录

- □ 每个文件有一个存放在磁盘索引节点区的索引节点，称为磁盘索引节点，它包括以下内容：
  1. 文件主标识符和同组用户标识符；
  2. 文件类型：是普通文件、目录文件、符号连接文件或特别文件（又分块设备文件或字符设备文件）；
  3. 文件主，同组用户和其它人对文件存取权限（读R、写W、执行X）；
  4. 文件的物理地址，用于UNIX直接、间接混合寻址的13个地址项di_addr[13]；
  5. 文件长度（字节数）di_size；
  6. 文件链接数di_nlink；
  7. 文件最近存取和修改时间等。

- □ UNIX采用文件名和文件说明分离的目录结构如下图所示：

UNIX的树型目录



I=1

bin	boot	unix	usr	mnt	etc	tmp	dey
2	3	4	5				

I=2

ls			
10			

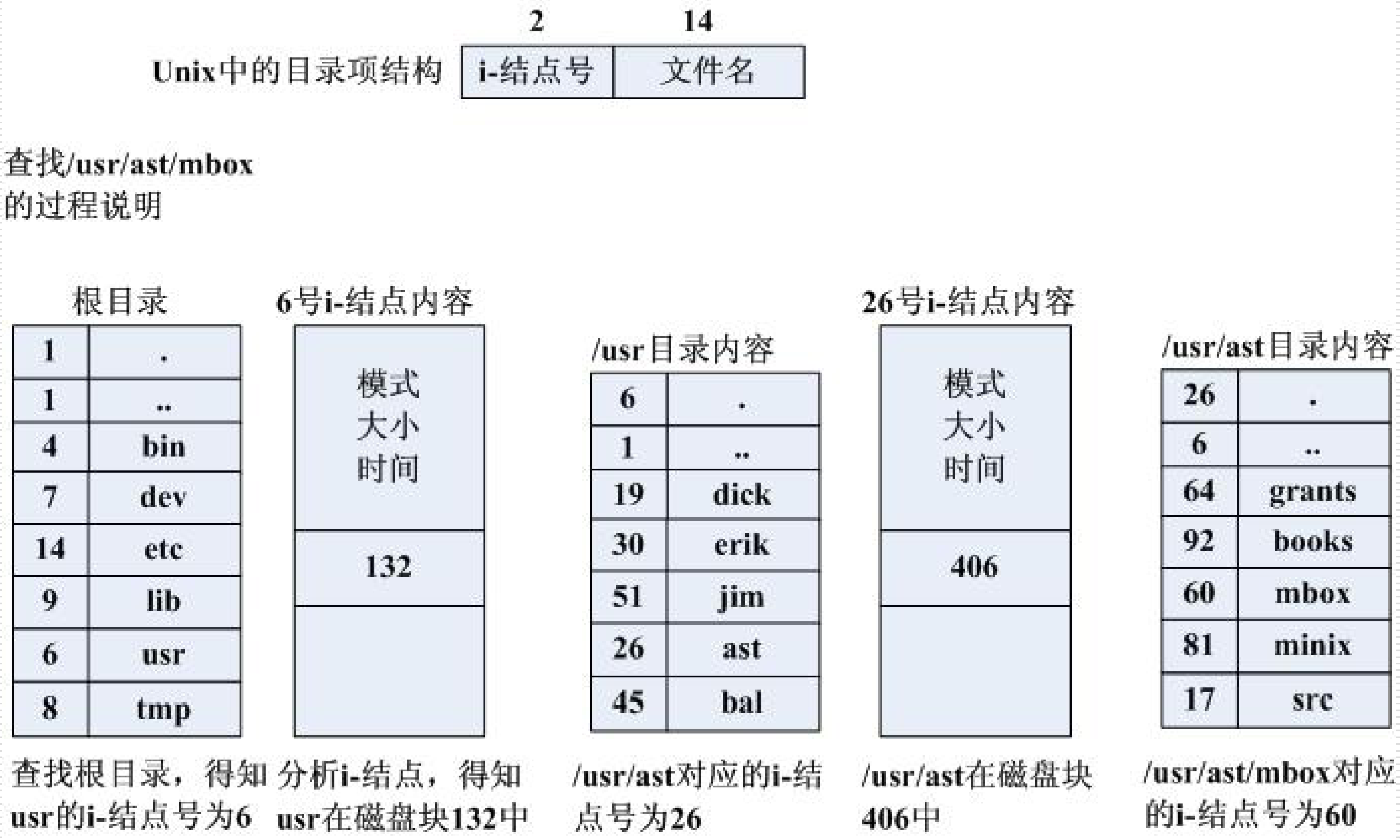
I=5

dir			
10			

I=10

ls

UNIX的树型目录



Unix中的目录项结构

2	14
i-结点号	文件名

查找/usr/ast/mbox
的过程说明

根目录

1	.
1	..
4	bin
7	dev
14	etc
9	lib
6	usr
8	tmp

查找根目录，得知usr的i-结点号为6

6号i-结点内容

模式 大小 时间
132

分析i-结点，得知usr在磁盘块132中

/usr目录内容

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast对应的i-结点号为26

26号i-结点内容

模式 大小 时间
406

/usr/ast在磁盘块406中

/usr/ast目录内容

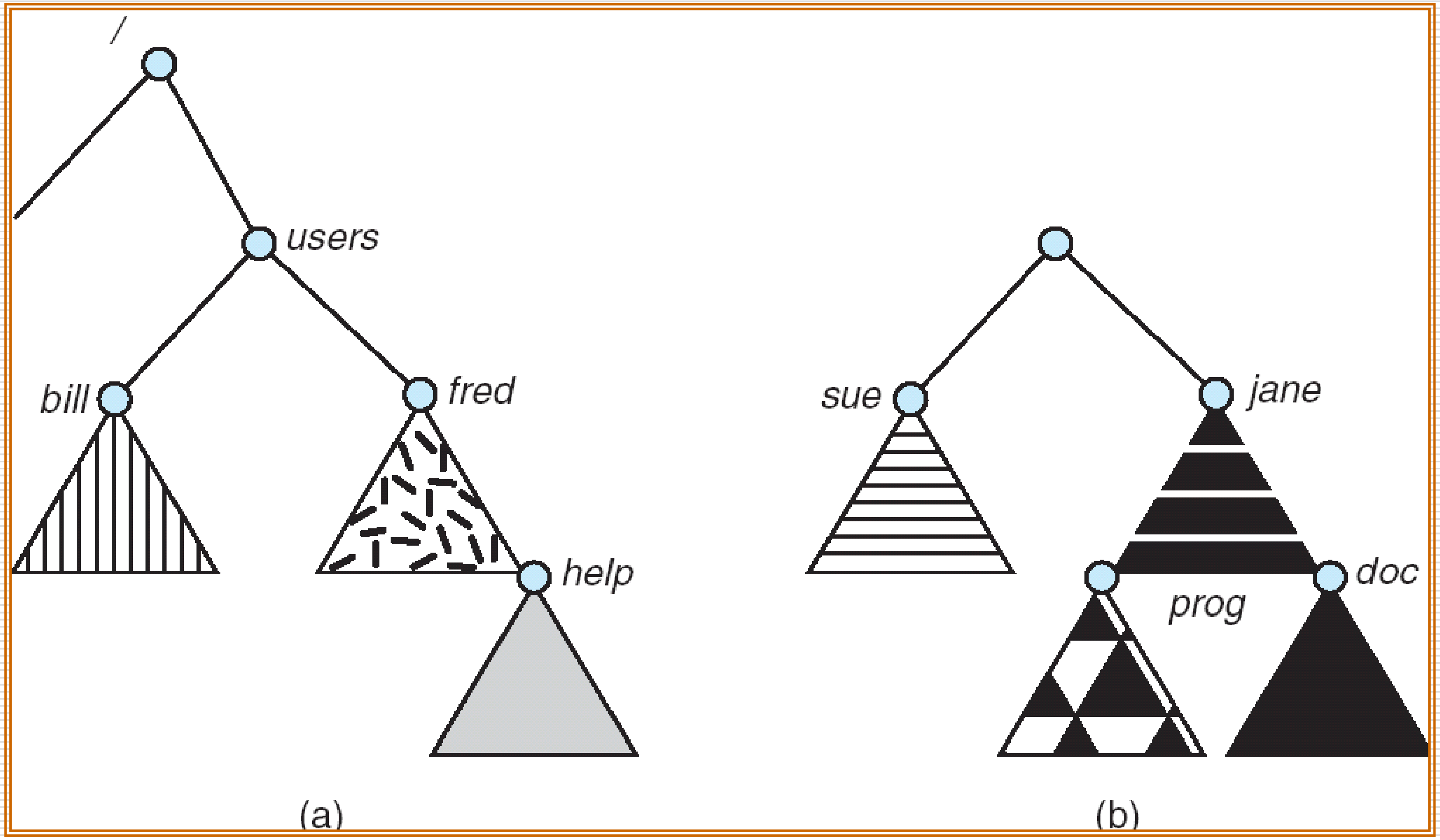
26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox对应的i-结点号为60

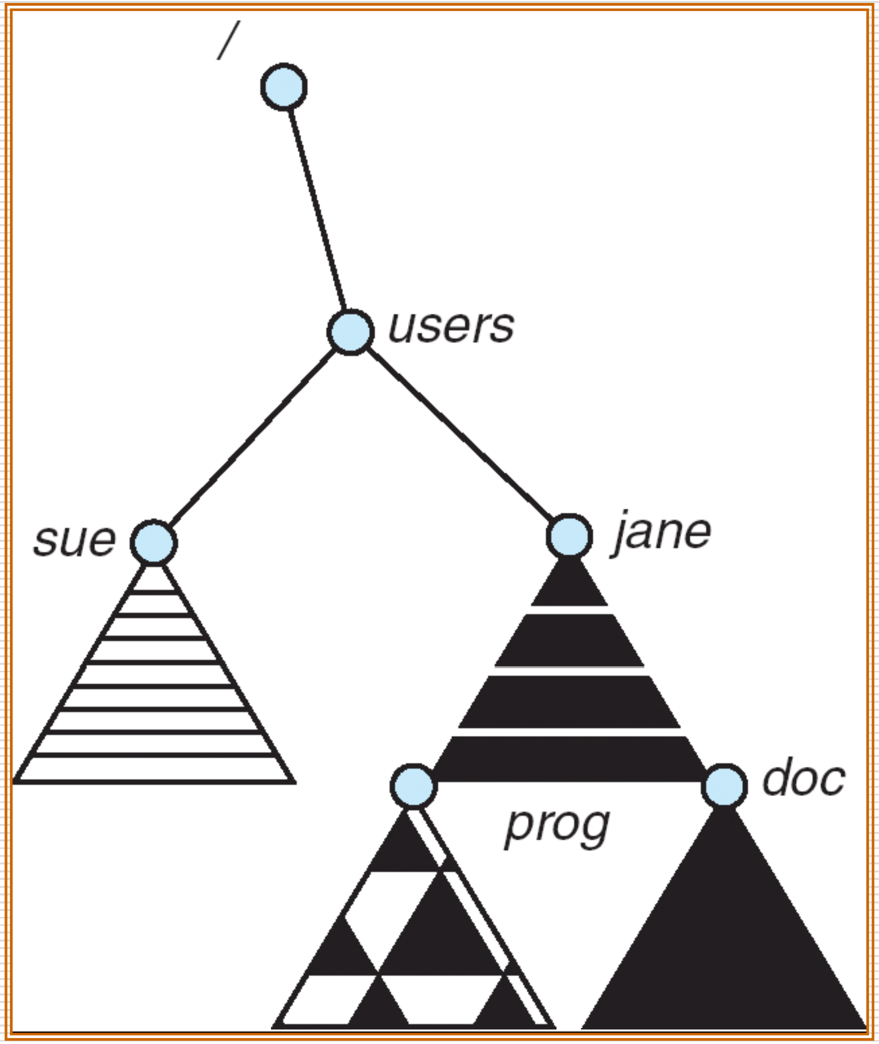
10.4 File System Mounting

- □ A file system must be **mounted** before it can be accessed
- □ A unmounted file system (i.e. Fig. 11-11(b)) is mounted at a **mount point**

(a) Existing. (b) Unmounted Partition



Mount Point



File System Mounting

- □ The operating system verifies that the device contains a valid file system.
- □ Asking the device driver to read the device directory and verifying that the directory has the expected format.
- □ The os notes in its directory structure that a file system is mounted at the specified mount point.

10.5 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

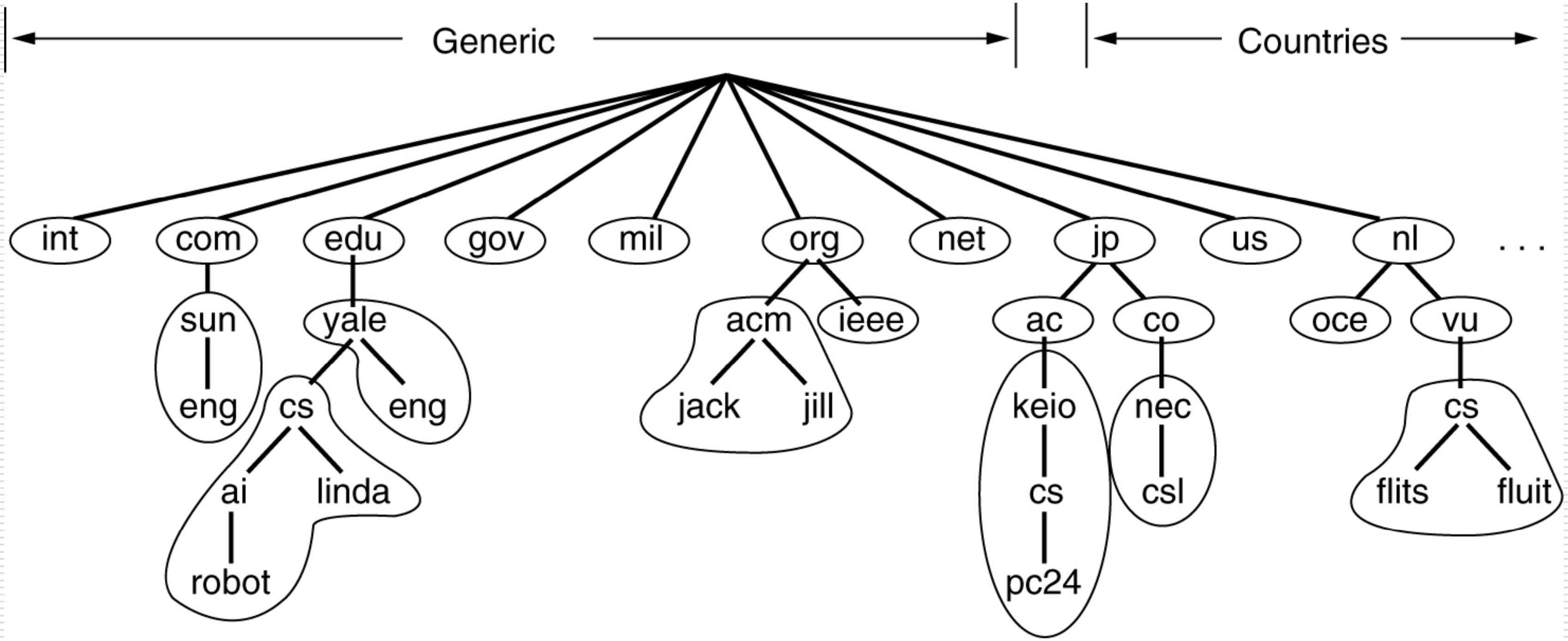
- □ File sharing, file naming and file protection are important
- □ Implement– maintain more file and directory attributes
- □ **User IDs** identify users, allowing permissions and protections to be per-user
  Group IDs allow users to be in groups, permitting group access rights

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

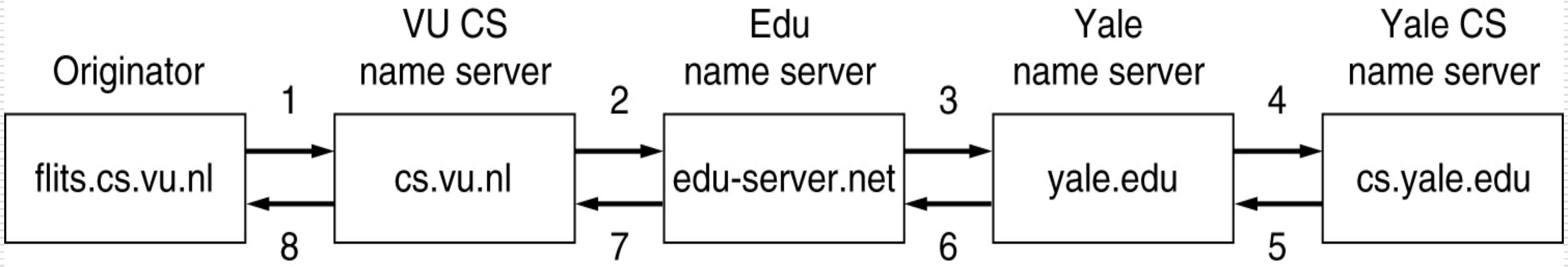
Name Servers

Part of the DNS name space showing the division into zones.



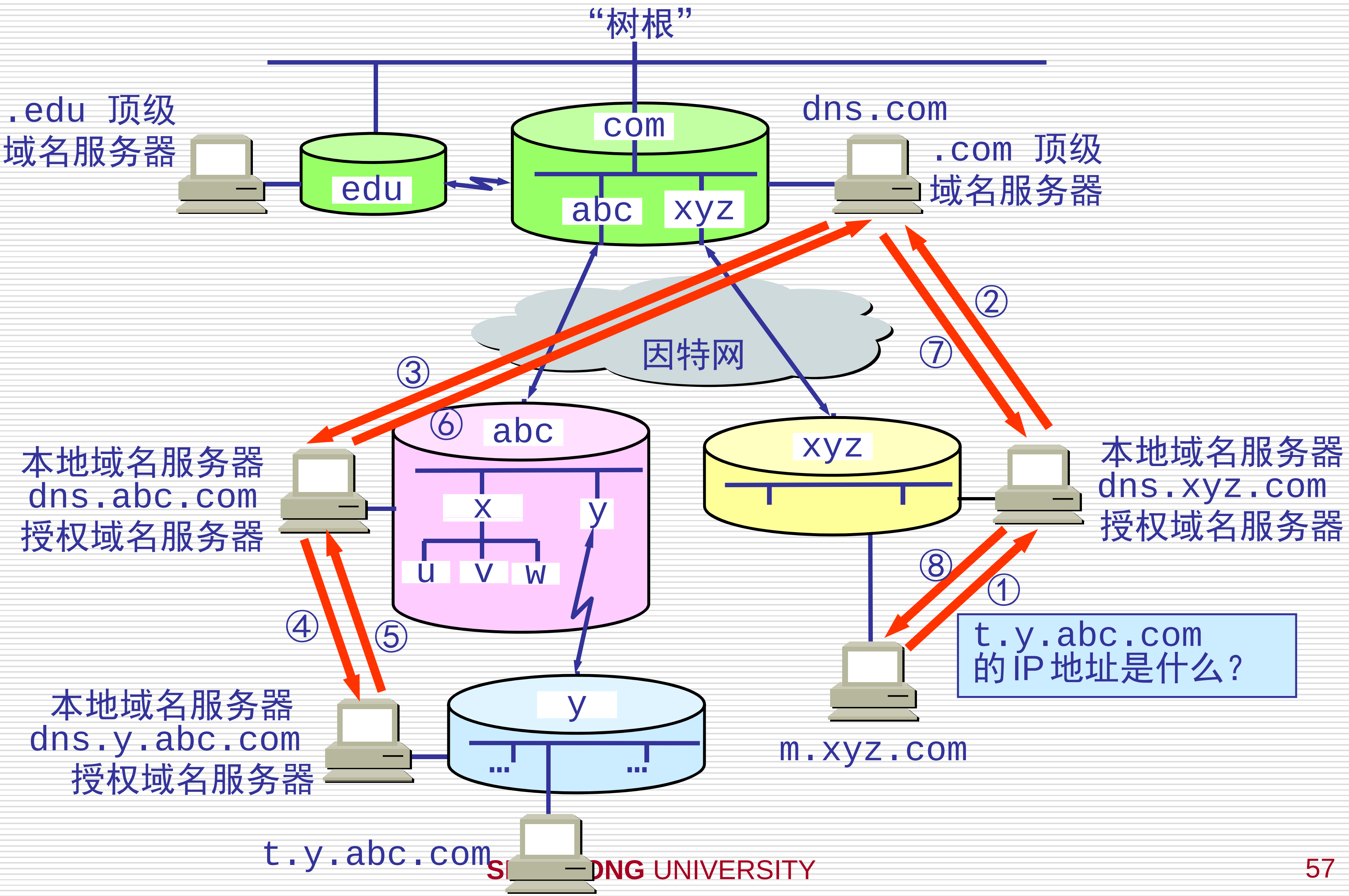
Name Servers (2)

How a resolver looks up a remote name in eight steps.



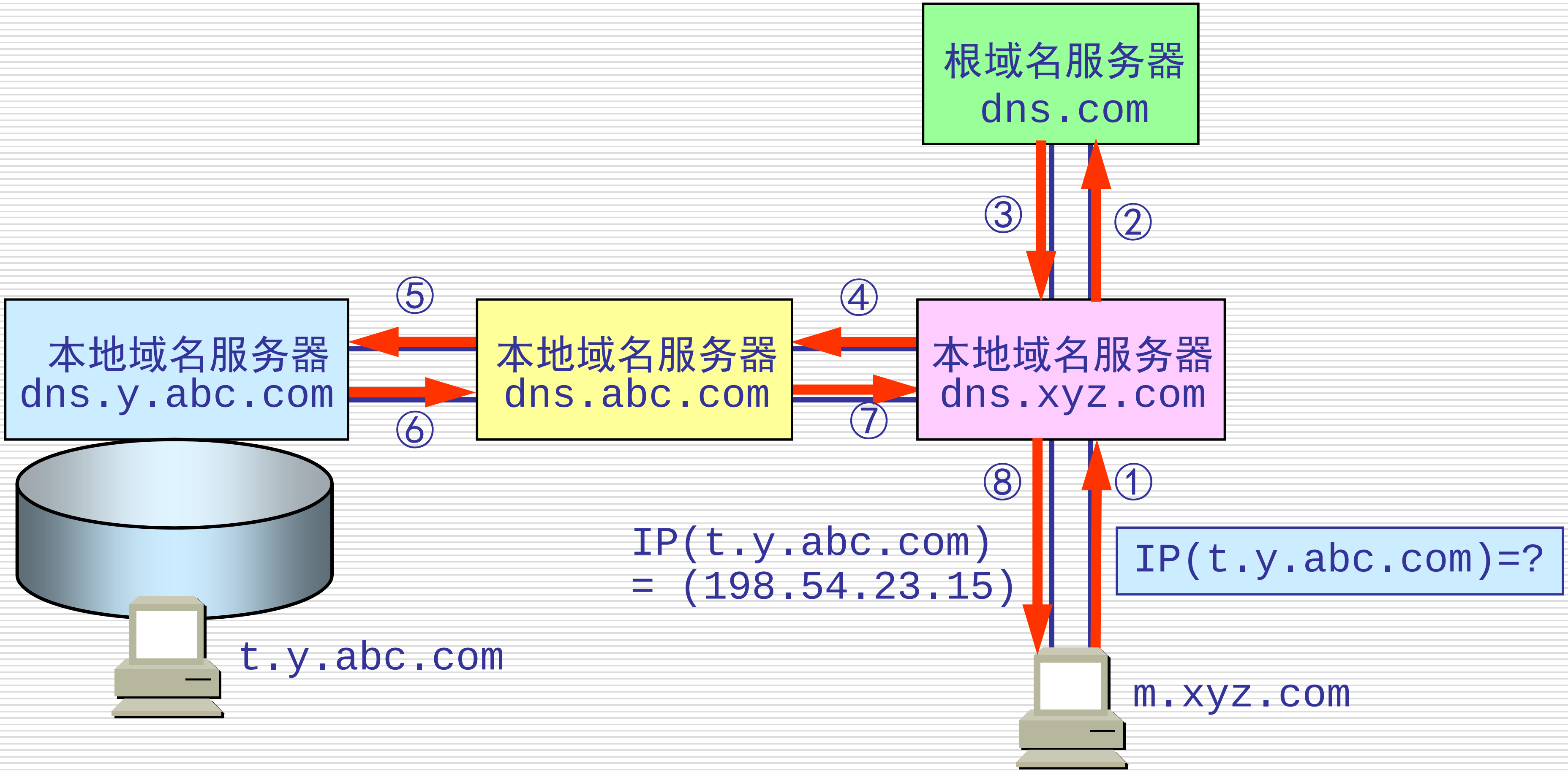
To find x.cs.yale.edu

递归查询



t.y.abc.com
的 IP 地址是什么？

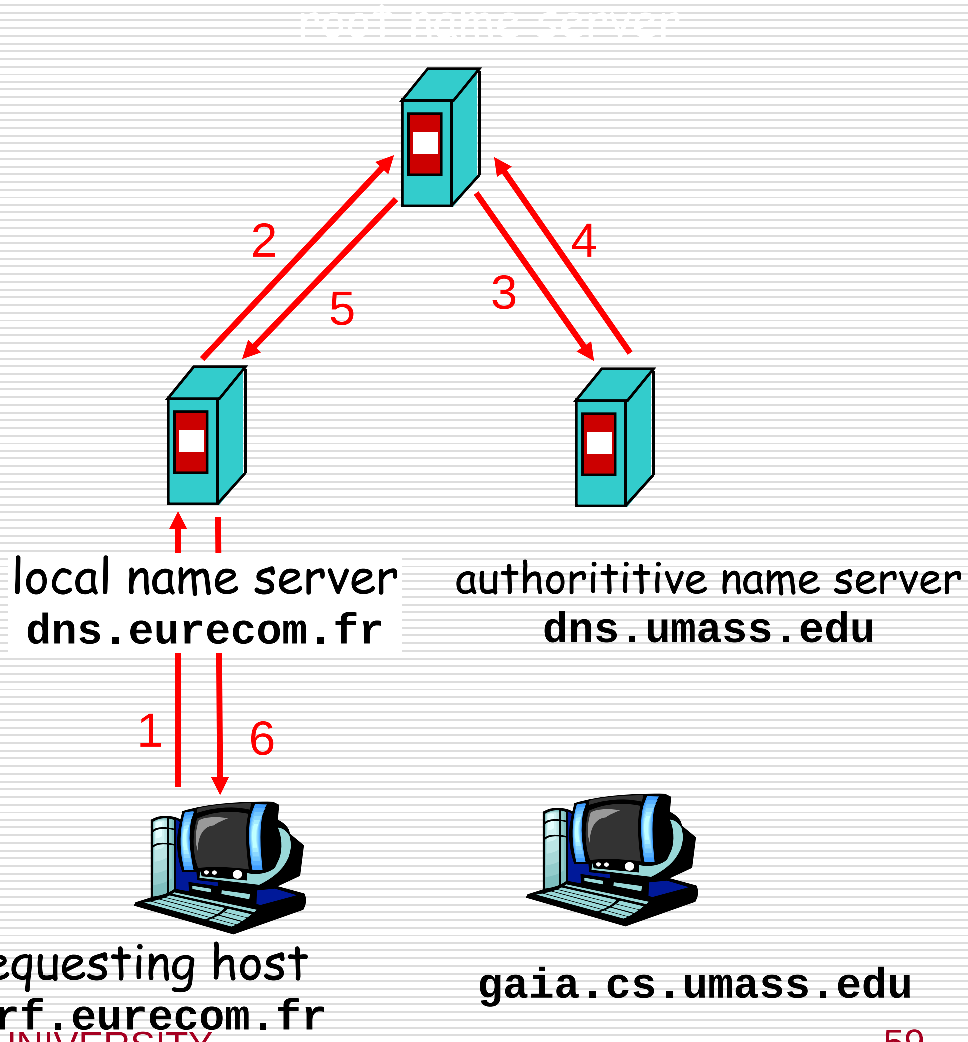
递归与迭代相结合的查询



Simple DNS example

host **`surf.eurecom.fr`** wants IP address of **`gaia.cs.umass.edu`**

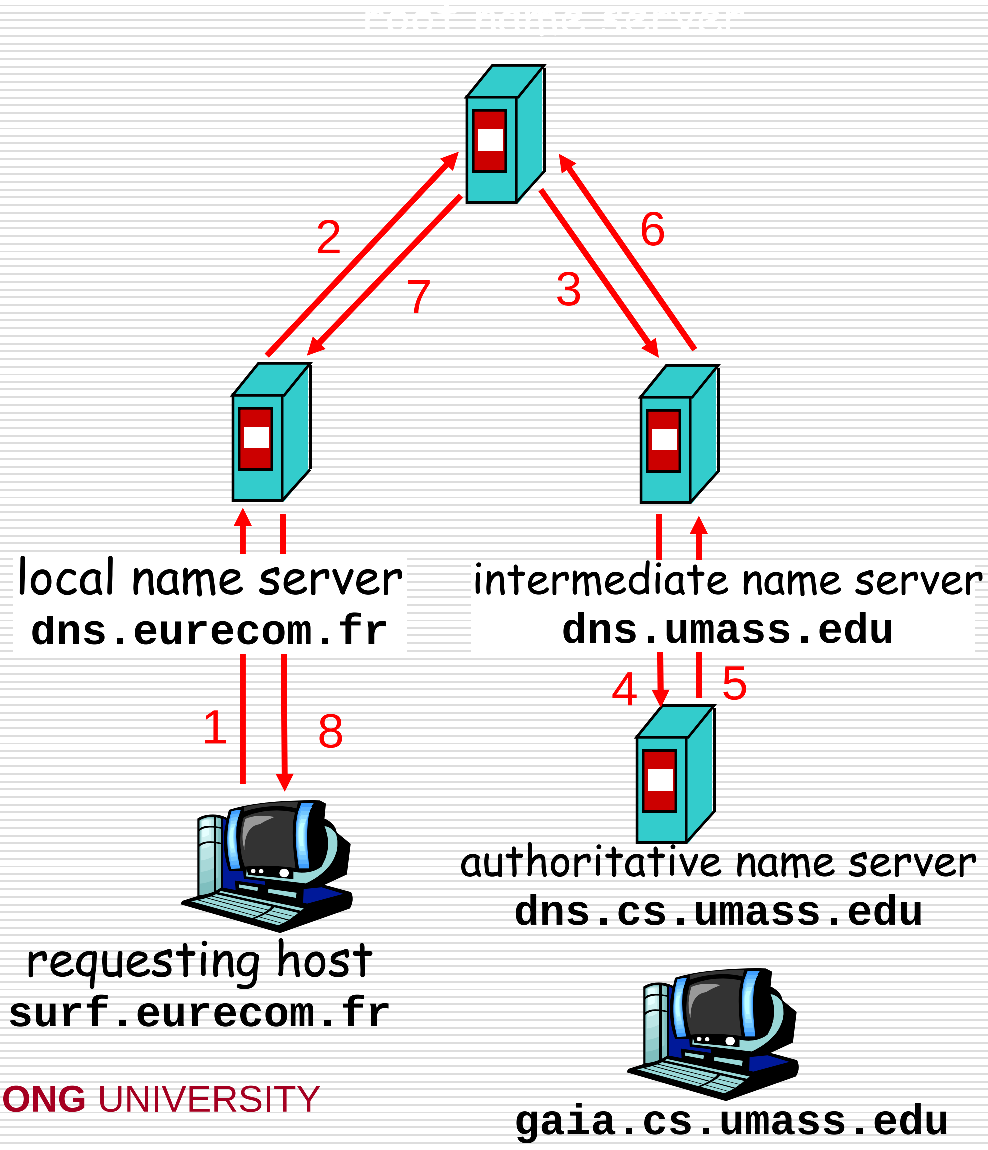
1. Contacts its local DNS server, **`dns.eurecom.fr`**
2. **`dns.eurecom.fr`** contacts root name server, if necessary
3. root name server contacts authoritative name server, **`dns.umass.edu,`** if necessary



DNS example

Root name server:

- may not know authoratiative name server
- may know *intermediate name server:* who to contact to find authoritative name server



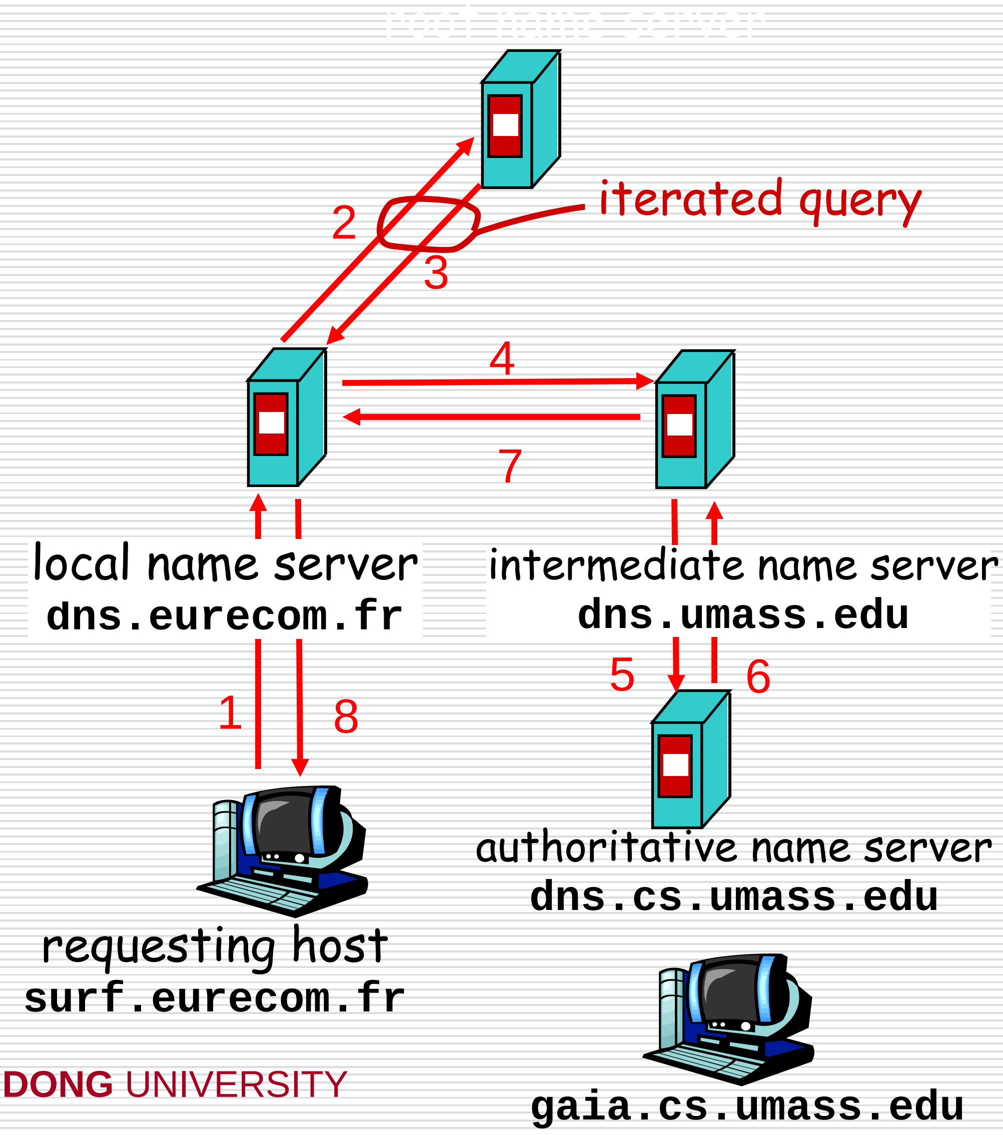
DNS: iterated queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



File Sharing – Failure Modes

- 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 include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- **Consistency semantics** specify how multiple users are to access a shared file simultaneously
  - Similar to Ch 7 process synchronization algorithms
    - Tend to be less complex due to disk I/O and network latency (for remote file systems
  - 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

10.6 Protection

- We want to keep file system safe from
  - Physical damage (reliability)
  - Improper access (protection)
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Single-user system
  - Simple：removing the floppy disks or locking them in a desk drawer
- Multi-user system
  - Complicated ➔

Types of access

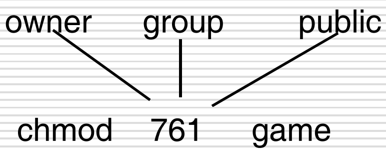
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List
- Other operations
  - Rename
  - Copy
  - Edit

Access Lists and Groups

- Associate each file or directory an access control list.
- Mode of access: read, write, execute
- Three classes of users

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.



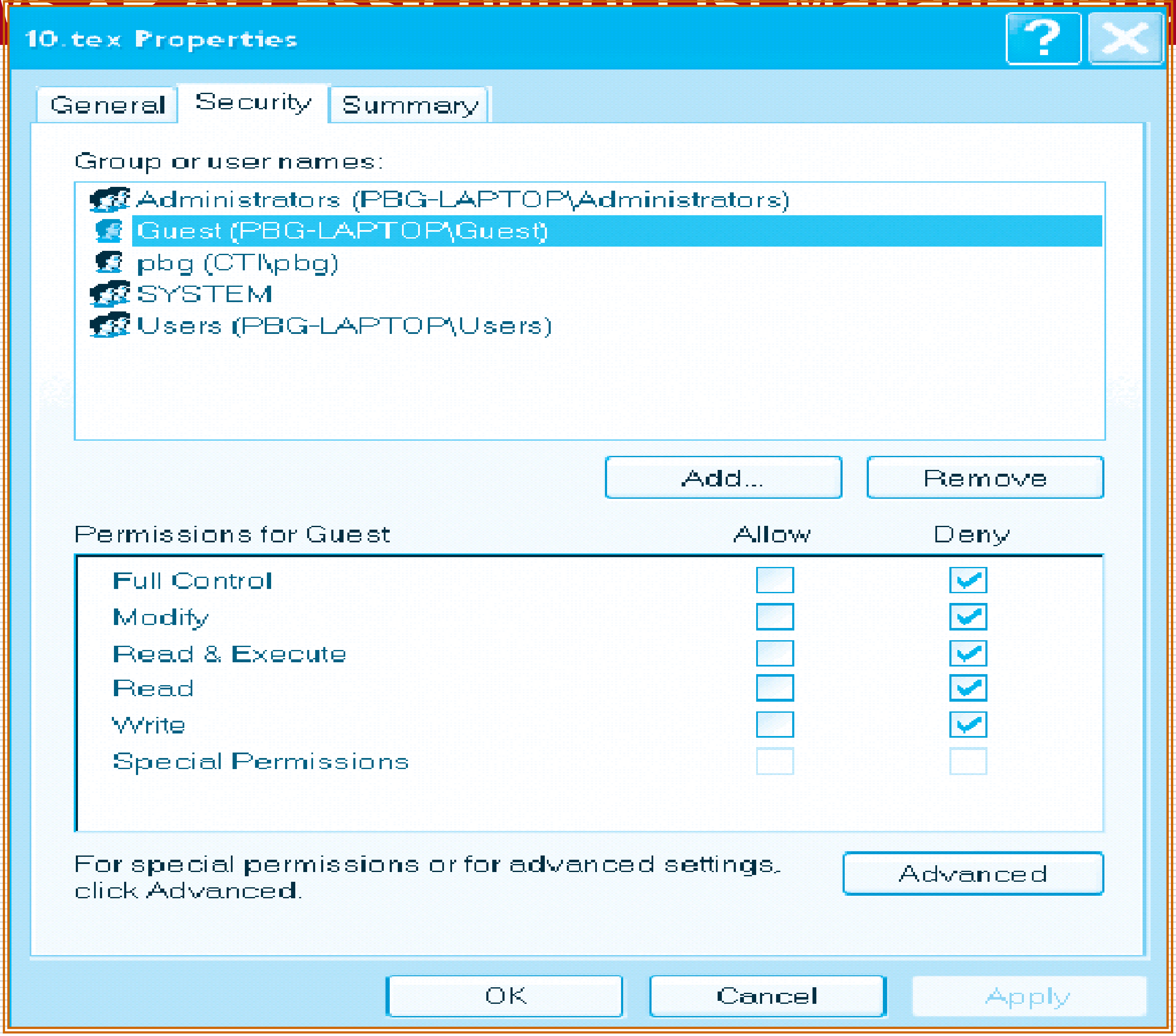
owner group public

chmod 761 game

Attach a group to a file

```
chgrp    G    game
```

Windows XP Access-control List Management



10.tex Properties

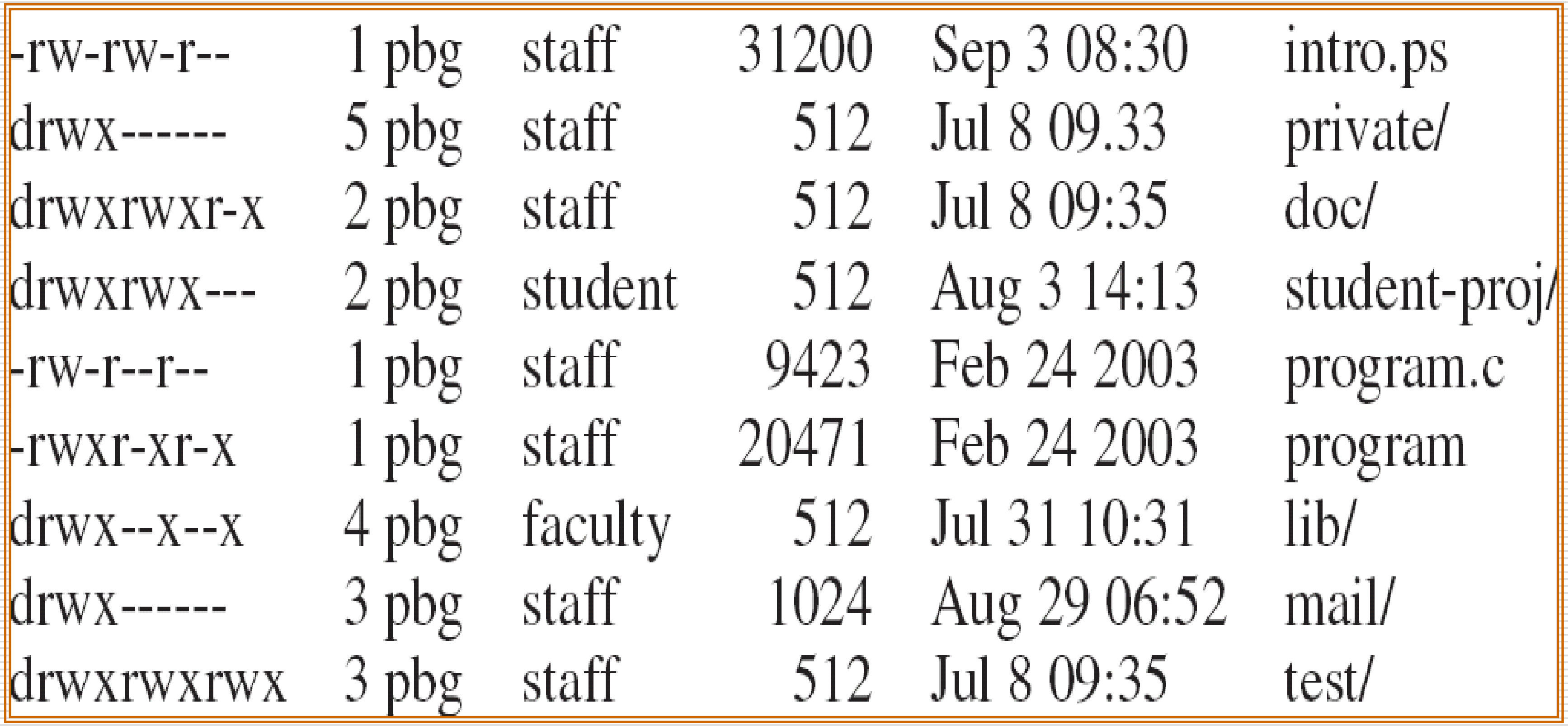
Group or user names:

- Administrators (PBG-LAPTOP\Administrators)
- Guest (PBG-LAPTOP\Guest)
- pbg (CTI\pbg)
- SYSTEM
- Users (PBG-LAPTOP\Users)

Permissions for Guest	Allow	Deny
Full Control	□	☑
Modify	□	☑
Read & Execute	□	☑
Read	□	☑
Write	□	☑
Special Permissions	□	□

For special permissions or for advanced settings, click Advanced.

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/
```

Other protection approaches

- □ A password with each file
- □ Directory protection

Assignment

□ 4, 10

End of Chapter 10

Any Question?