



# 操作系统原理

## Operating Systems Principles

陈鹏飞  
计算机学院



中山大學  
SUN YAT-SEN UNIVERSITY

计算机学院（软件学院）

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



# 第十三讲 — 文件系统





中山大學

SUN YAT-SEN UNIVERSITY

计算机学院 (软件学院)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



# 目标

- 文件系统的功能;
- 文件系统接口;
- 文件系统的设计权衡, 访问方法、共享、加锁以及目录结构;
- 文件系统保护;



# 文件

- ❖ 操作系统对存储设备的物理属性加以抽象，从而定义逻辑存储单位，即文件，文件由操作系统映射到物理设备上。
- ❖ 用户创建的数据集合；
- ❖ 对于用户来说，文件系统是操作系统中最重要的部分之一；

## Long-term existence

- files are stored on disk or other secondary storage and do not disappear when a user logs off

## Sharable between processes

- files have names and can have associated access permissions that permit controlled sharing

## Structure

- files can be organized into hierarchical or more complex structure to reflect the relationships among files



# 文件概念

## ❖ Contiguous logical address space

## ❖ Types:

### ■ Data

- Numeric
- Character
- Binary

### ■ Program

## ❖ Contents defined by file's creator

### ■ Many types

- **text file,**
- **source file,**
- **executable file**



# 文件属性

- ❖ **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**





中山大學

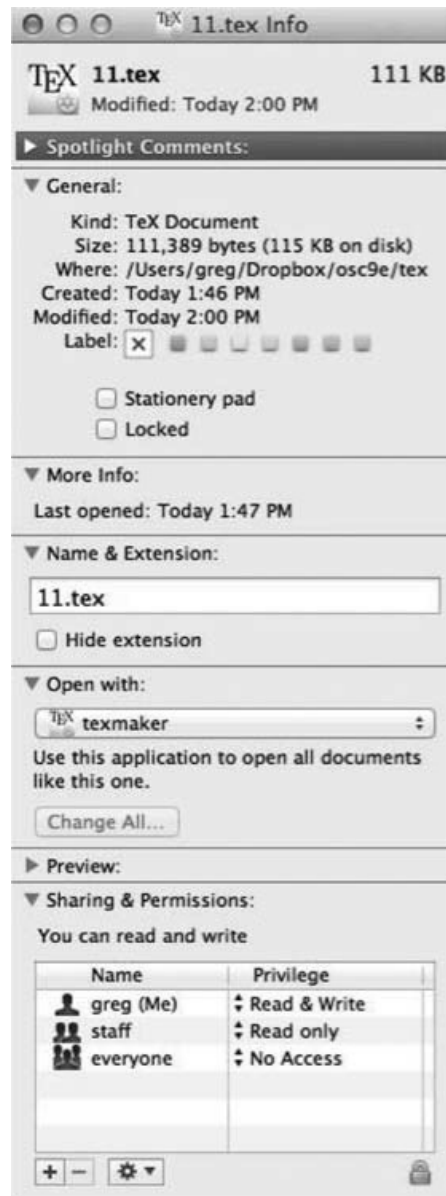
SUN YAT-SEN UNIVERSITY

计算机学院 (软件学院)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



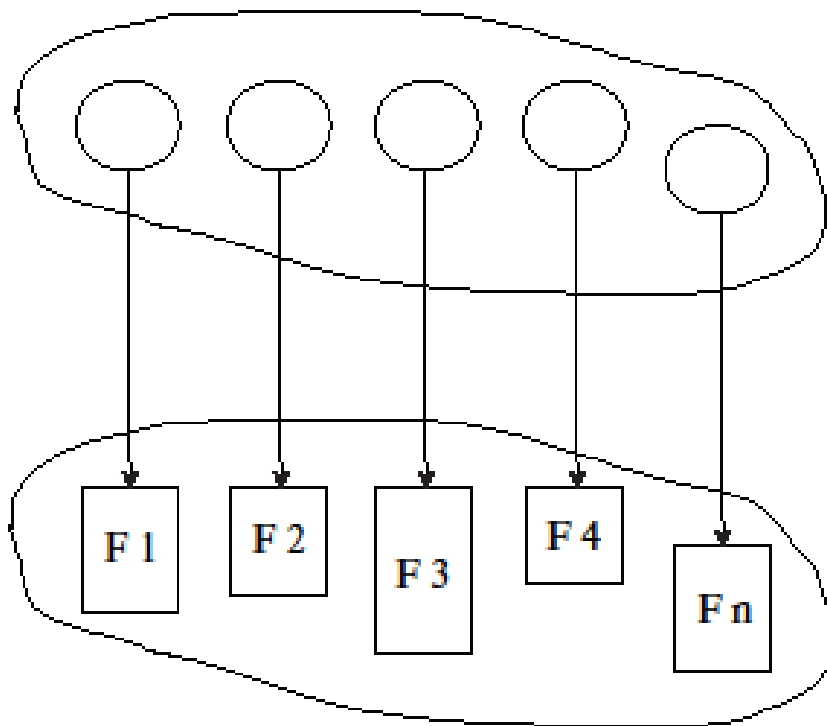
# Mac OS上的文件属性





# 文件目录结构

- ❖ A collection of nodes containing information about all files



- ❖ Both the directory structure and the files reside on disk





# 文件操作

- ❖ Create
- ❖ Write – at write pointer location
- ❖ Read – at read pointer location
- ❖ Reposition within file - seek
- ❖ Delete
- ❖ Truncate (截断)
- ❖ *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

这6个基本操作组所需  
文件操作的最小集合



# 文件打开 (Open)

- ❖ **Several pieces of data are needed to manage open files:**
  - **Open-file table:** tracks open files (进程打开表、系统打开表)
  - **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 (访问权限) :** per-process access mode information



# 文件锁

## ❖ Provided by some operating systems and file systems

- Similar to reader-writer locks
- **Shared lock**（共享锁） similar to reader lock – several processes can acquire concurrently
- **Exclusive lock**（互斥锁） similar to writer lock

## ❖ Mediates access to a file

## ❖ **Mandatory**（强制性的） or **advisory**（建议性的）：

- **Mandatory** – access is denied depending on locks held and requested (Windows)
- **Advisory** – processes can find status of locks and decide what to do (Unix)



## 文件锁案例——Java

```
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();
```



## 文件锁案例——Java

```
// 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();
}
}
```



# 文件类型

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

Windows通过扩展名识别文件类型， Unix系统通过文件开始部分的magic数字来识别类型





# 文件结构

- ❖ **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



# 文件访问方法

- ❖ A file is fixed length logical records
- ❖ Sequential Access
- ❖ Direct Access
- ❖ Other Access Methods

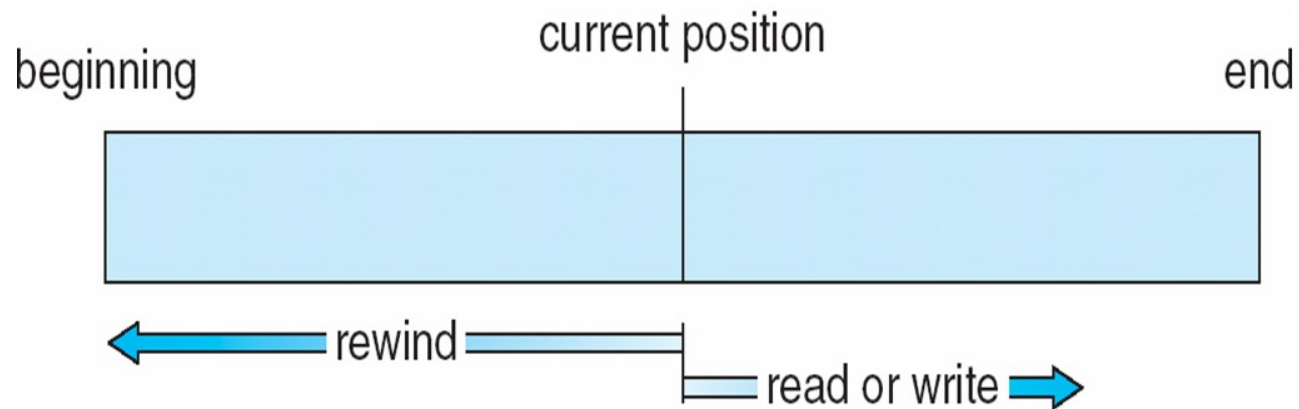


# 顺序访问

## ❖ Operations

- read next
- write next
- Reset

## ❖ Figure





## 直接访问

❖ **Operations**      适用于由固定长度的逻辑记录组成的文件;

- **read  $n$**
- **write  $n$**
- **position to  $n$** 
  - **read next**
  - **write next**
  - **rewrite  $n$**

**$n$  = relative block number**

❖ **Relative block numbers (相对块号) allow OS to decide where file should be placed**



## 直接访问模拟顺序访问

sequential access	implementation for direct access
<i>reset</i>	$cp = 0;$
<i>read next</i>	$read\ cp;$ $cp = cp + 1;$
<i>write next</i>	$write\ cp;$ $cp = cp + 1;$



## 其他访问方法

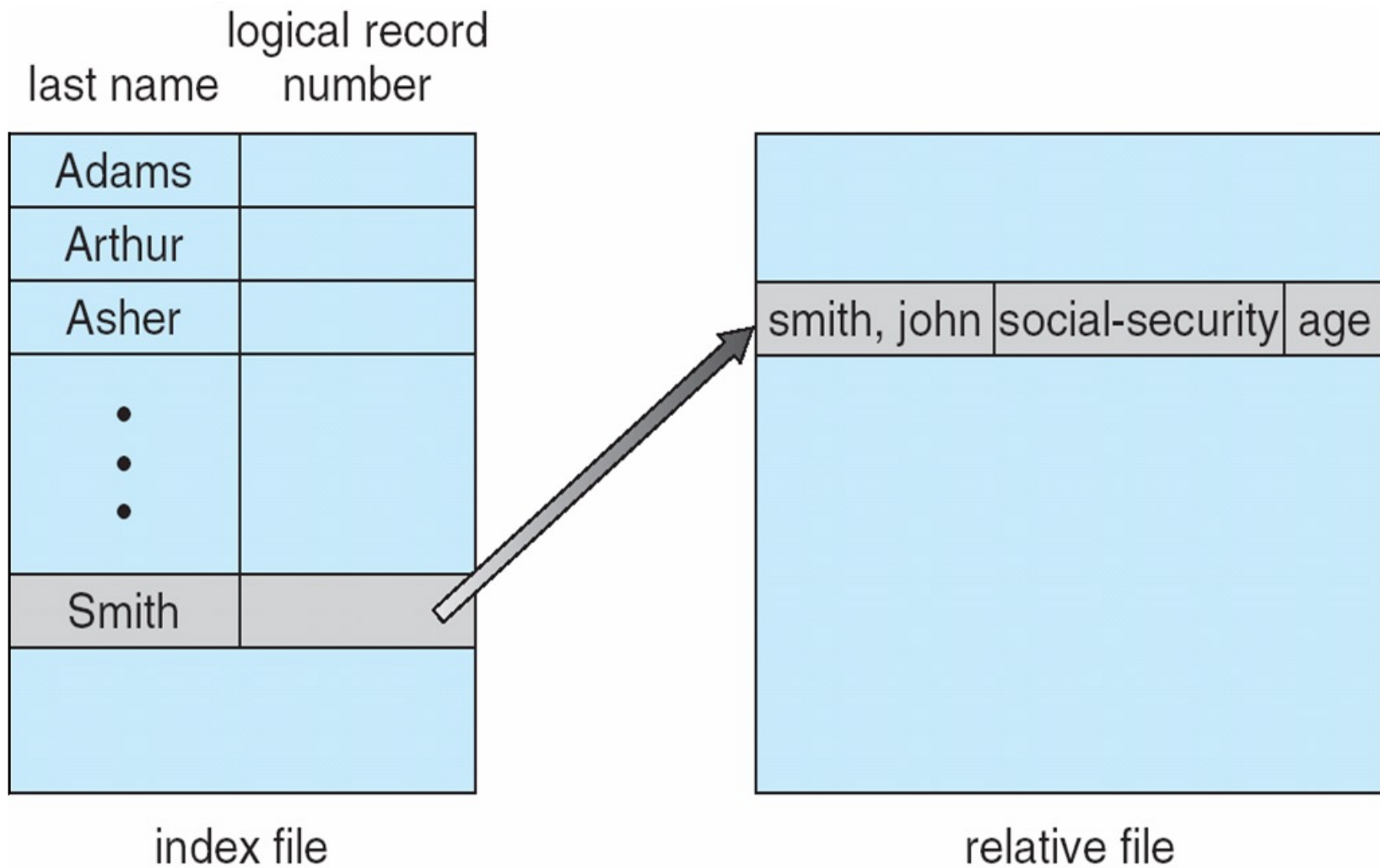
- ❖ Can be other access methods built on top of base methods
- ❖ General involve creation of an index for the file
- ❖ Keep index in memory for fast determination of location of data to be operated on (consider Universal Produce Code (UPC code) plus record of data about that item)
- ❖ If the index is too large, create an in-memory index, which an index of a disk index
- ❖ IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- ❖ VMS operating system provides index and relative files as another example (see next slide)

通过记录的键至多两次的直接访问就可以定位记录





# 索引文件和相关文件的例子



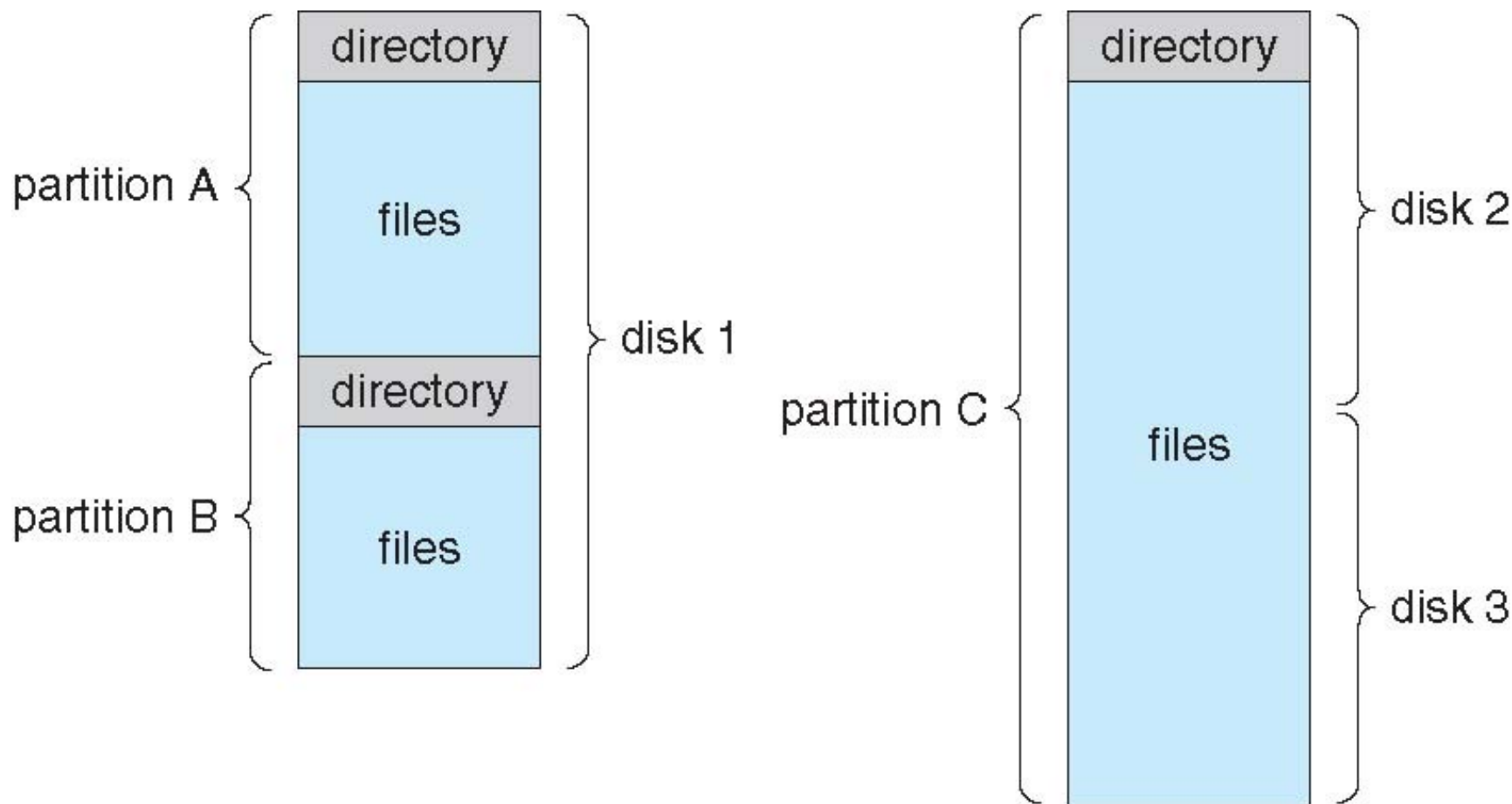


## 磁盘结构

- ❖ Disk can be subdivided into partitions
- ❖ Disks or partitions can be RAID protected against failure
- ❖ Disk or partition can be used raw – without a file system, or formatted with a file system
- ❖ Partitions also known as minidisks, slices
- ❖ Entity containing file system is known as a volume(卷)
- ❖ Each volume containing a file system also tracks that file system's info in device directory or volume table of contents
- ❖ In addition to general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer



# 存储结构





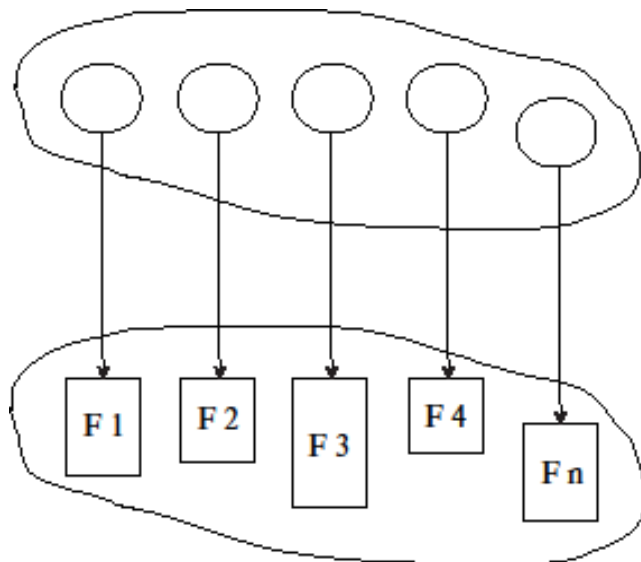
# 文件系统的类型

- ❖ We mostly talk of general-purpose file systems
- ❖ But systems frequently have many file systems, some general- and some special- purpose
- ❖ Consider Solaris has
  - tmpfs – memory-based volatile FS for fast, temporary I/O
  - objfs – interface into kernel memory to get kernel symbols for debugging
  - ctfs – contract file system for managing daemons
  - lofs – loopback file system allows one FS to be accessed in place of another
  - procfs – kernel interface to process structures
  - ufs, zfs – general purpose file systems



## 目录结构

- ❖ A collection of nodes containing information about all files
- ❖ 符号表，可以将文件名称转成目录条目



- ❖ Both the directory structure and the files reside on disk



# 目录结构

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





# 目录结构

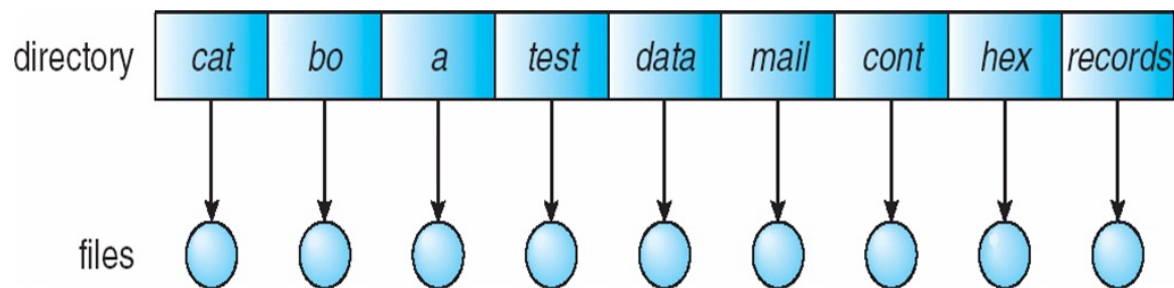
The directory is organized logically to obtain

- ❖ **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, ...)**



# 单级目录

❖ A single directory for all users



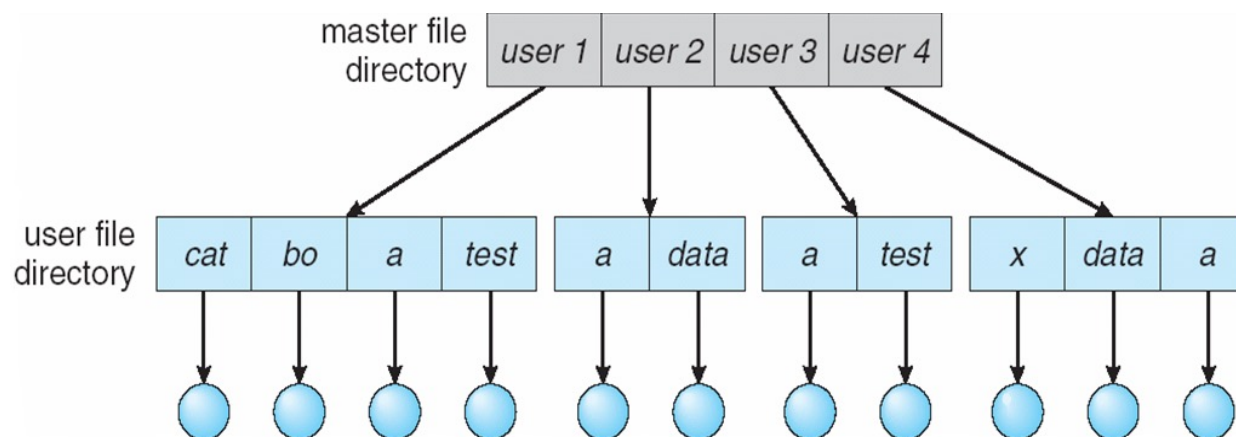
❖ Naming problem

❖ Grouping problem



## 两级目录结构

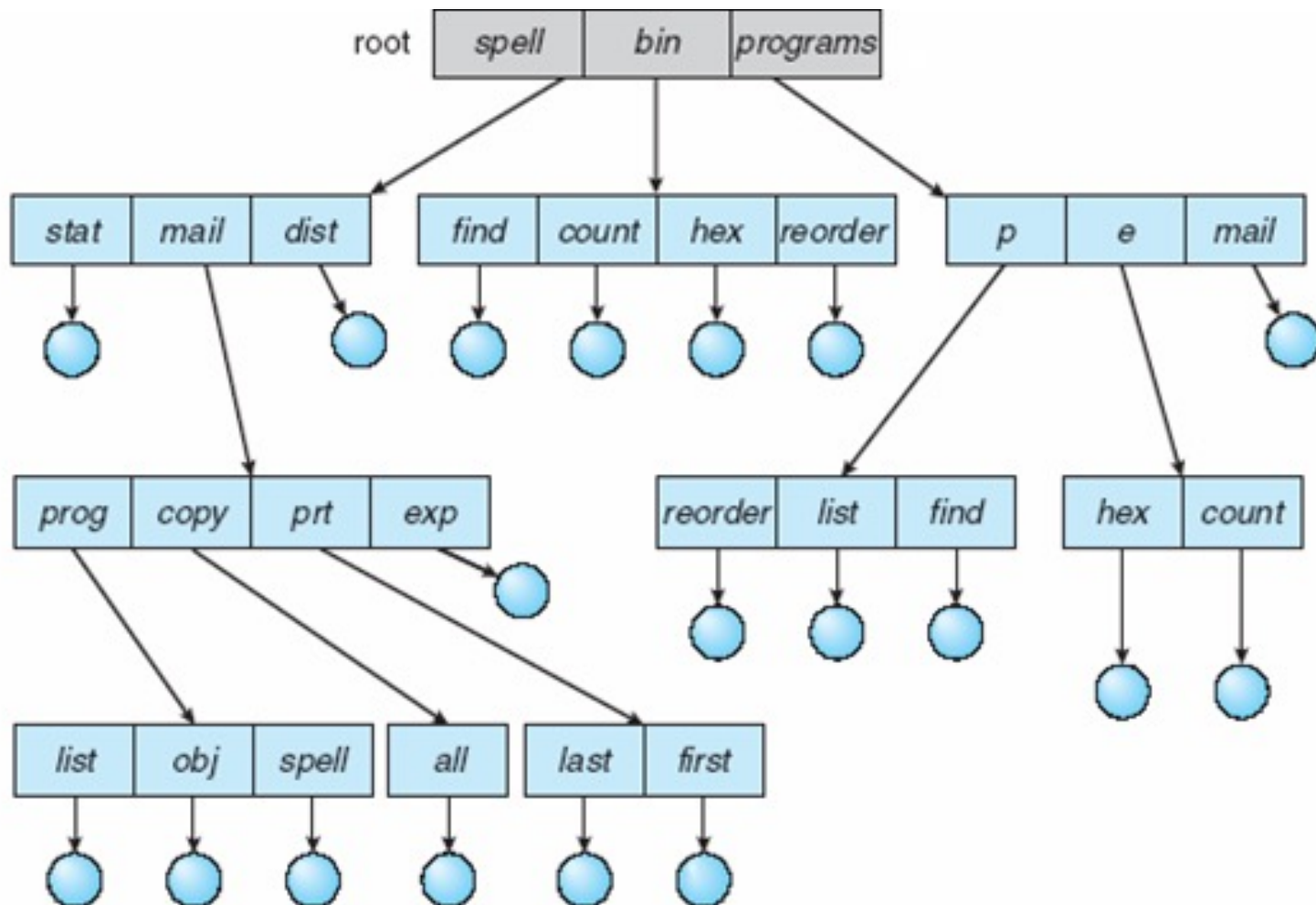
### ❖ Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching



# 树形目录

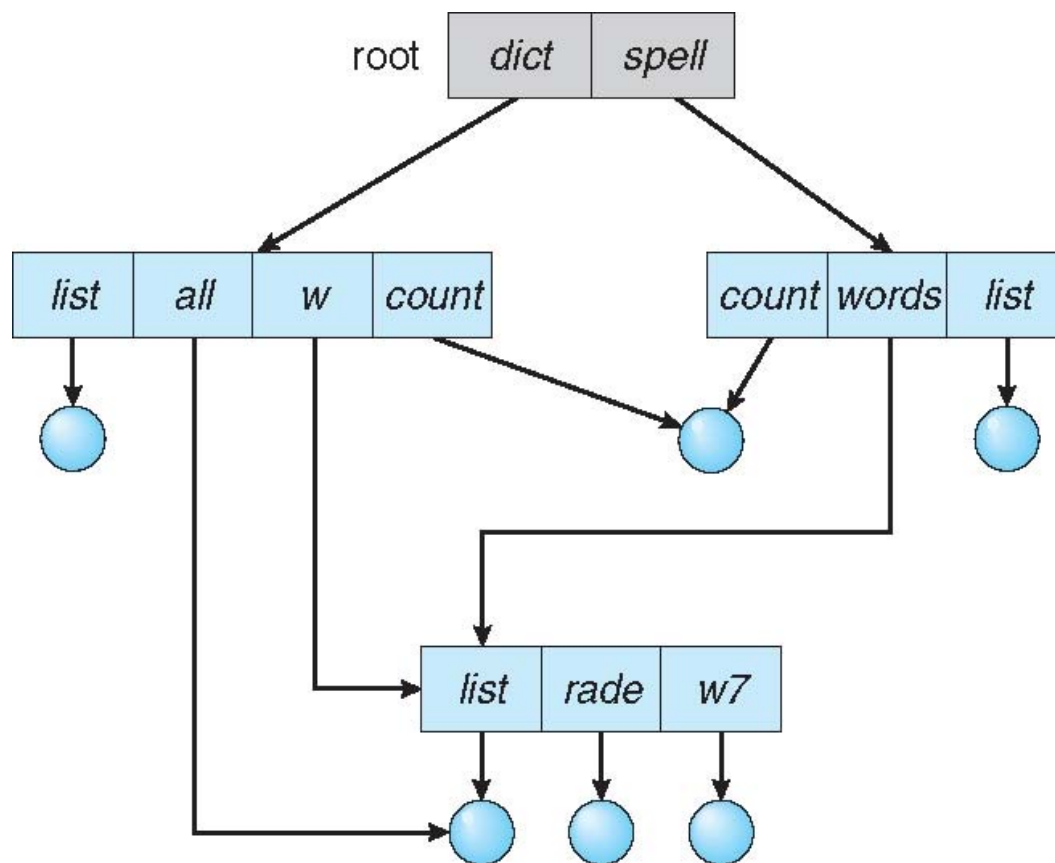




# 无环图目录

❖ Have shared subdirectories and files

❖ Example





# 无环图目录

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

## Solutions:

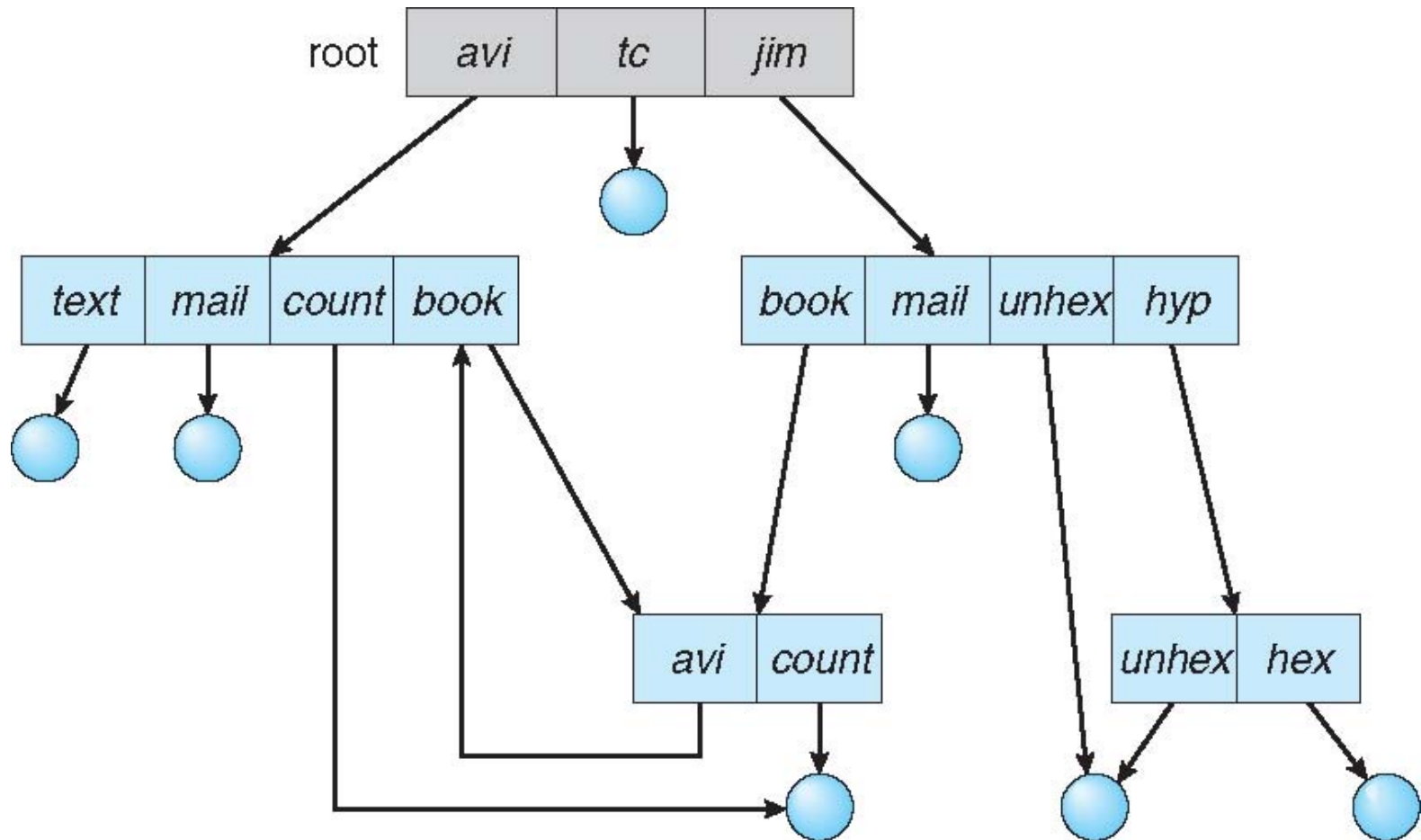
- Backpointers, so we can delete all pointers.
  - Variable size records a problem
- Backpointers using a daisy chain organization
- Entry-hold-count solution
- ❖ New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

软链接、硬链接





# 通用图目录





# 通用图目录

## ❖ How do we guarantee no cycles?

- Allow only links to files not subdirectories
- **Garbage collection**
- Every time a new link is added use a cycle detection algorithm to determine whether it is OK

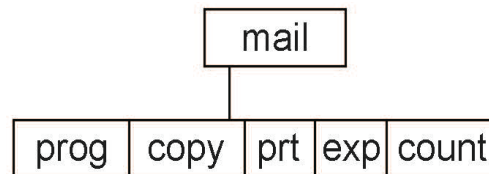


## 当前目录

- ❖ Can designate one of the directories as the current (working) directory
  - `cd /spell/mail/prog`
  - `type list`
- ❖ Creating and deleting a file is done in current directory
- ❖ Example of creating a new file
  - If in current directory is `/mail`
  - The command

**`mkdir <dir-name>`**

- Results in:



- Deleting “mail”  $\Rightarrow$  deleting the entire subtree rooted by “mail”



# 文件系统保护

## ❖ File owner/creator should be able to control:

- What can be done
- By whom

## ❖ Types of access

- Read
- Write
- Execute
- Append
- Delete
- List



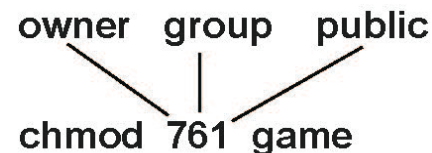
# Linux文件访问

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

			<b>RWX</b>
a) owner access	7	⇒	1 1 1
			<b>RWX</b>
b) group access	6	⇒	1 1 0
			<b>RWX</b>
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 file (say *game*) or subdirectory, define an appropriate access.

chgrp G game



- ❖ Attach a group to a file 37



## Unix文件目录

-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/

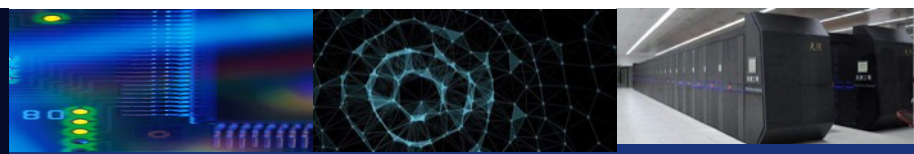




中山大學  
SUN YAT-SEN UNIVERSITY

计算机学院 (软件学院)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



# 谢谢