文件系统界面 (File System Interface)

“文件”只是个逻辑概念

- 文件，表示一段连续的逻辑地址空间
- 文件类型
  - 数据文件
    - 数字文件
    - 文本文件
    - 二进制文件
  - 程序

文件类型 – 以文件扩展名区分

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

文件结构

- 无结构文件 – 一串字节流
- 简单的记录型结构
  - 行文件
  - 固定长度文件
  - 变长度文件
- 复杂结构
  - 格式化文档
  - 重定位可执行文件

文件结构（续）

- 可以在“无结构”文件 中插入控制字符，就模拟了其它有结构类型的文件。例如， Linux 文件
- 那么，谁决策的？
  - 操作系统？
  - 应用程序？

文件属性 (Attributes)

- **文件名** – 这是唯一一项以可行性指导文件信息
- **文件标识** – 在文件系统内部，文件的唯一身份标志 (数字)
- **文件类型** – OS 用于支持各种类型文件
- **位置** – 文件内容在存储设备的首地址
- **长度** – 当前文件长度
- **保护** – 控制用户的读、写、执行等权限
- **时间，日期，用户标识** – 文件的保护、安全、监控操作时用到的数据
- 关于文件的属性信息不是文件内容本身。保存在文件存储空间的目录结构中。

文件操作

- 创建文件， Create
- 写， Write
- 读， Read
- 文件内定位， lseek
- 删除， Delete
- 截取， Truncate
- *Open*(F_i) – 在目录结构中搜寻文件 F_i，并且将其文件属性复制到内存
- *Close* (F_i) – 将文件 F_i 的文件属性，从内存写回到磁盘的目录结构

文件访问方式

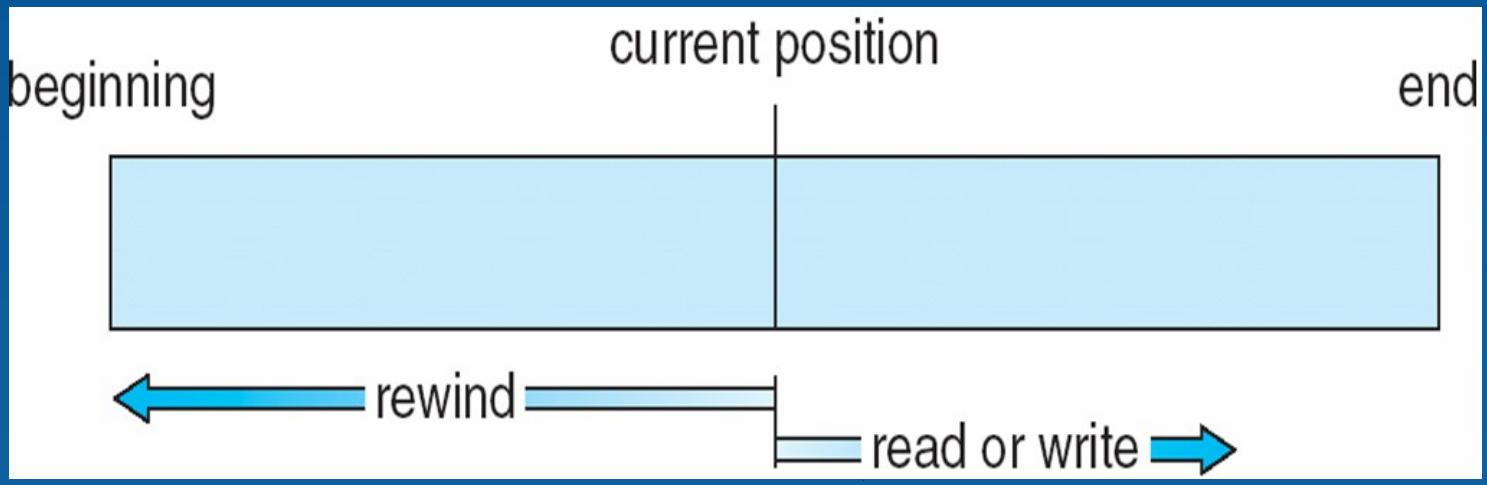
- 直接访问
  - read *n*
  - write *n*
  - position to *n*
    - read next
    - write next
  - rewrite *n*

n = 相对数据块号

文件访问方式(续)

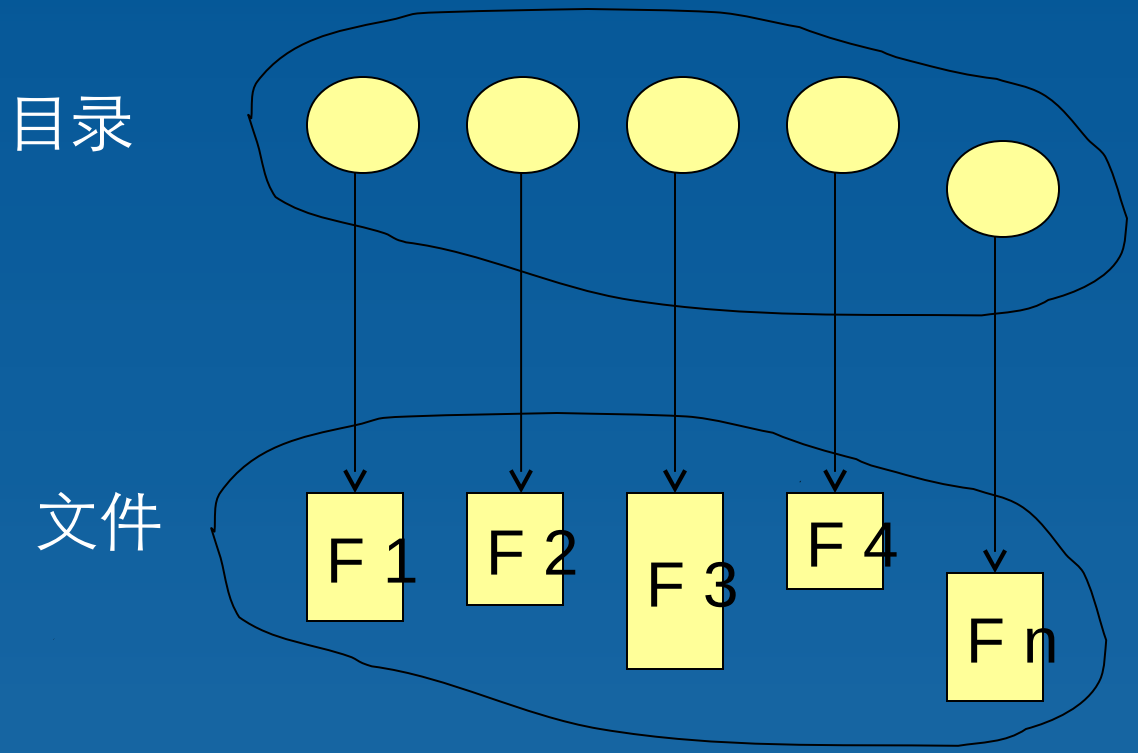
- 串行访问
  - read next
  - write next
  - reset

串行访问文件



目录结构

- 一堆节点的集合。节点包含了关于文件的信息



目录结构驻留在磁盘。文件也驻留在磁盘

针对目录的操作

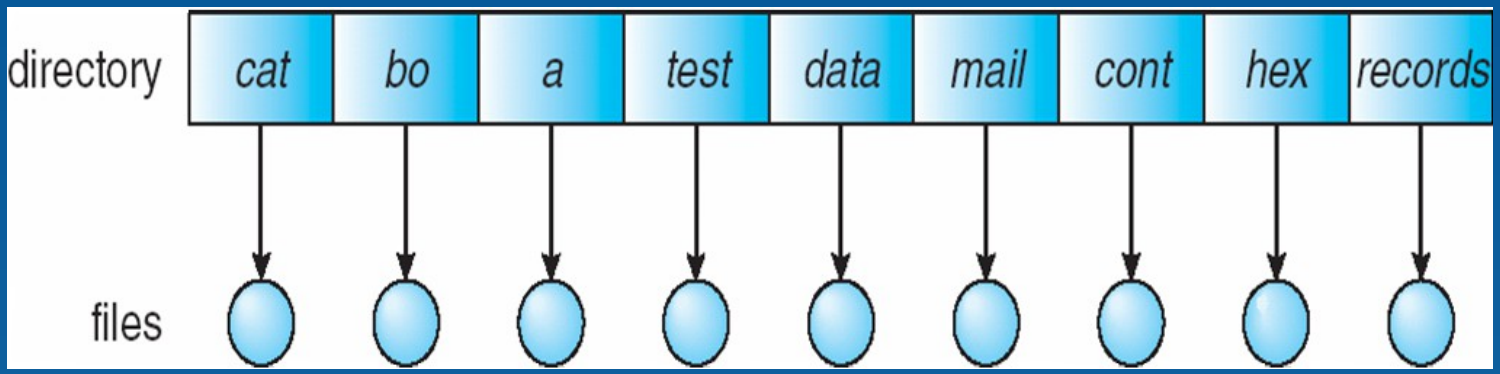
- 搜索文件
- 创建一个文件
- 删除文件
- 列表显示目录下的文件和子目录
- 文件改名
- 遍历文件系统

目录的逻辑组织优化

- 效率 – 快速定位文件
- 命名 – 方便用户
  - 两个用户以同一个名字命名不同的文件
  - 同一个文件可以有不同的名字
- 成组操作 – 将文件属性相同的文件，组合起来 (e.g., Java 程序，游戏软件 …)

单层目录结构

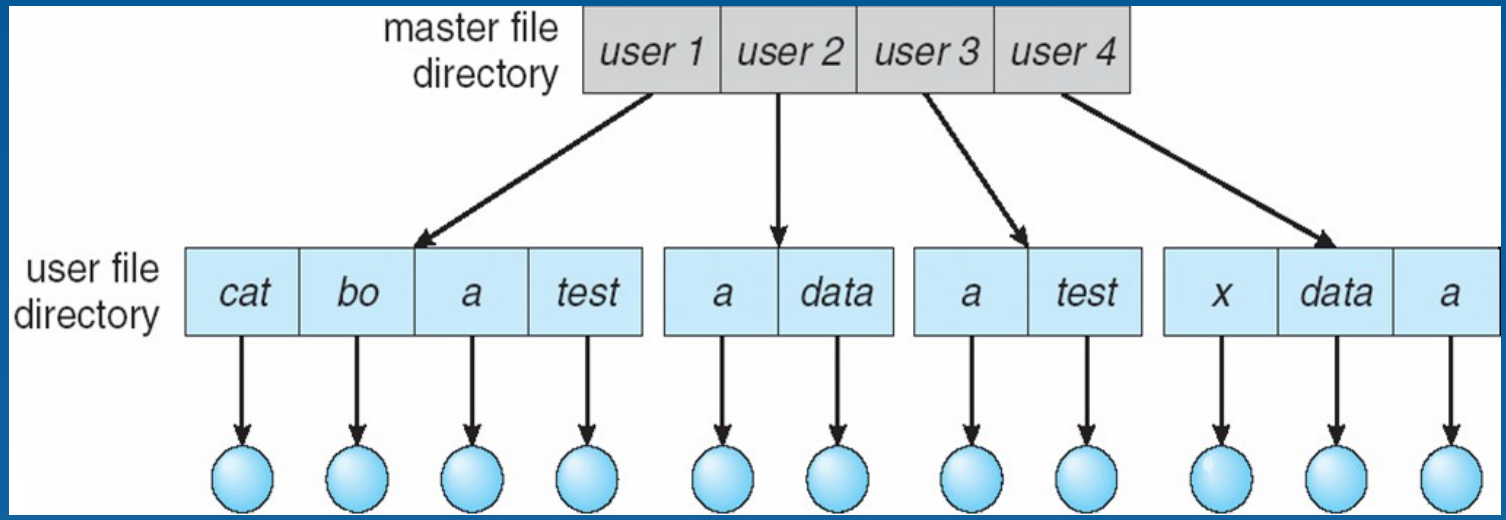
- 所有用户的所有文件，都放置在同一层



- 命名问题
- 不方便成组

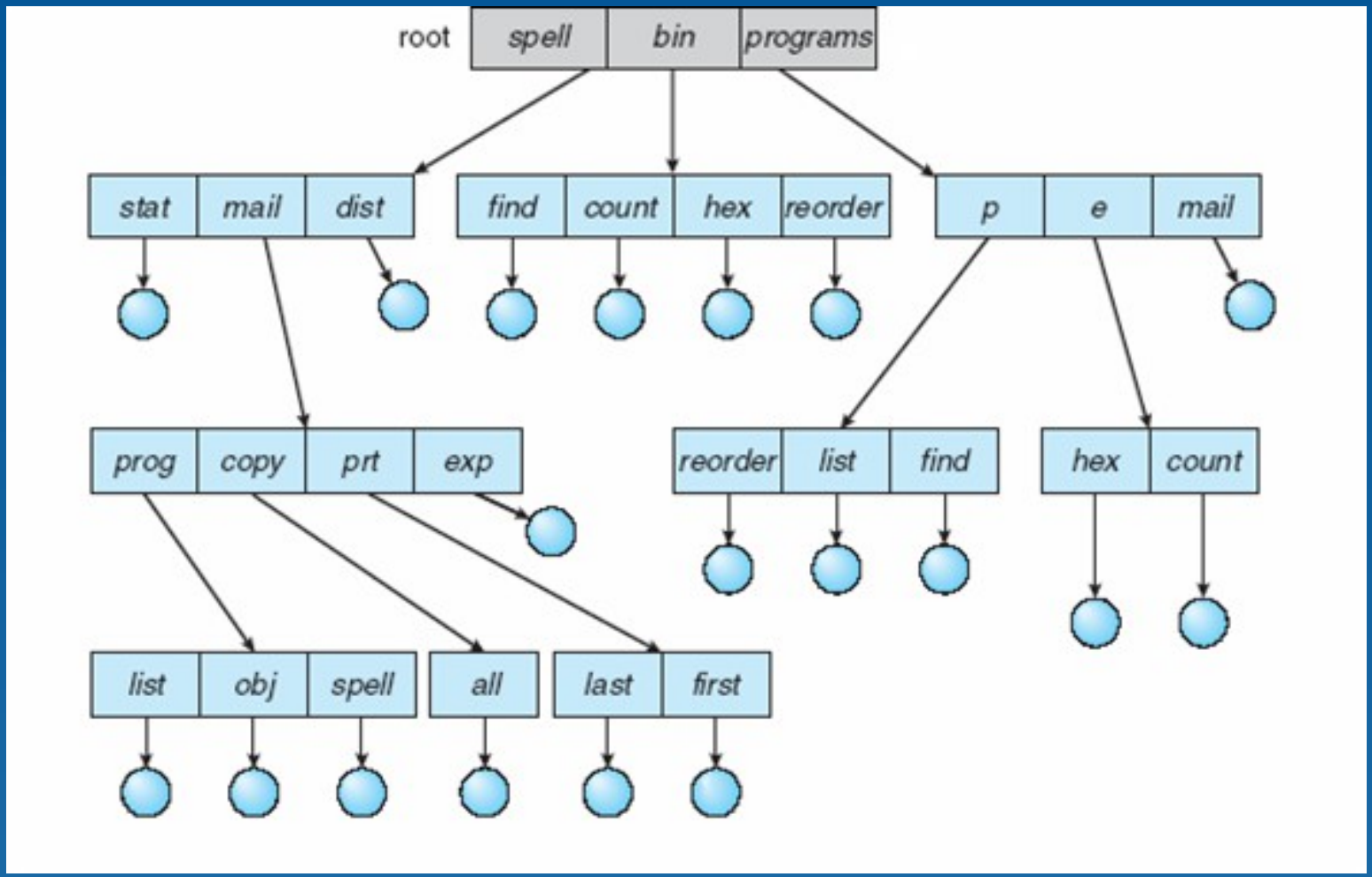
二层目录结构

- 每个用户有一个独立的目录



- 以路径名定位文件
- 不同的用户可以用相同的文件名，表示各自的文件
- 搜索效率提高
- 不能够成组操作

树型目录结构


root
spell
bin
programs
stat
mail
dist
find
count
hex
reorder
p
e
mail
prog
copy
prt
exp
reorder
list
find
hex
count
list
obj
spell
all
last
first

树型目录结构（续）

- 高效搜索
- 成组：把相关的文件放置在同一个子目录
- 当前目录（工作目录）概念
- **绝对路径，相对路径**
- 创建新文件，就是在当前目录创建一个新文件

树型目录结构（续）

- 删除一个文件，就是在当前目录删除指定文件

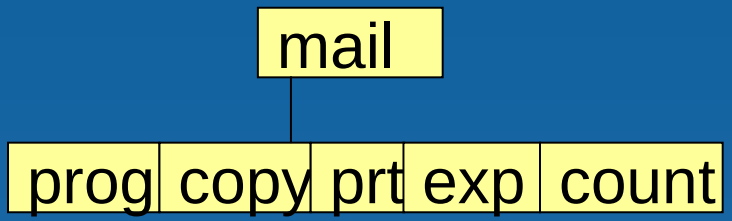
  rm <file-name>

- 创建子目录，就是在当前目录创建一个新子目录

  mkdir <dir-name>

  举例：假设当前目录是 /mail

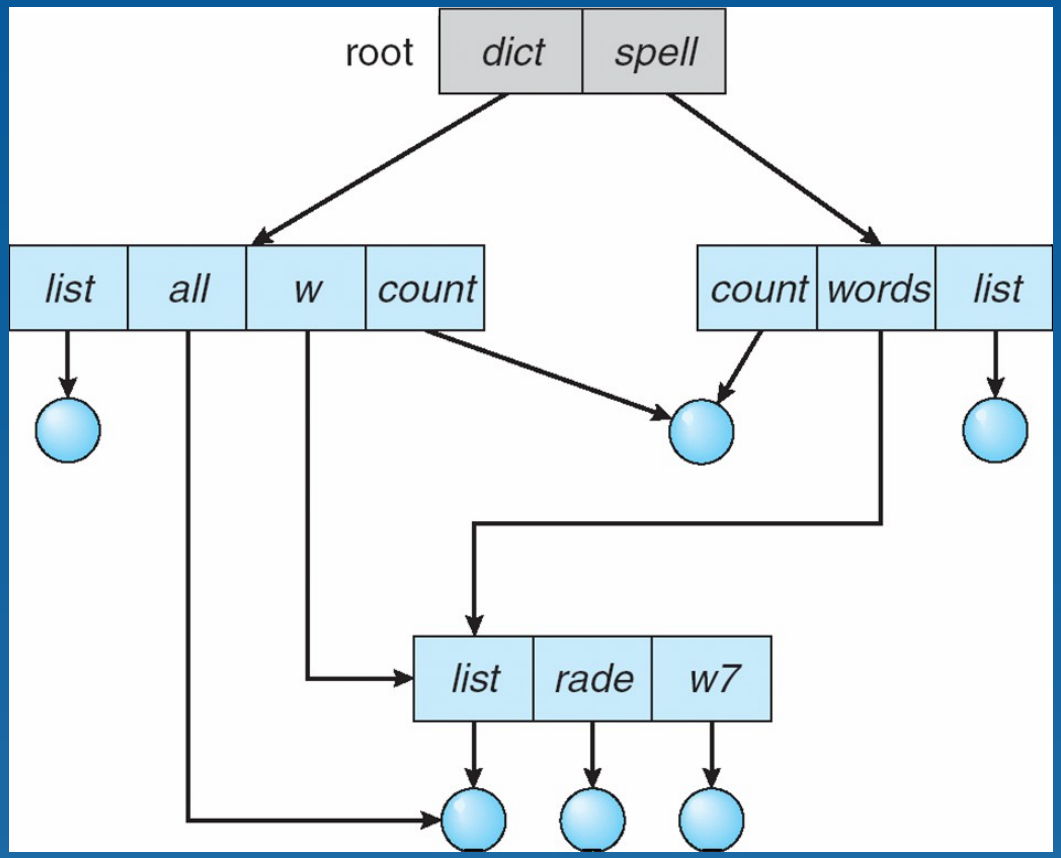
  mkdir count



删除 “mail” ⇒ 删除了 “mail” 以下的整个子目录

有向无环图目录结构

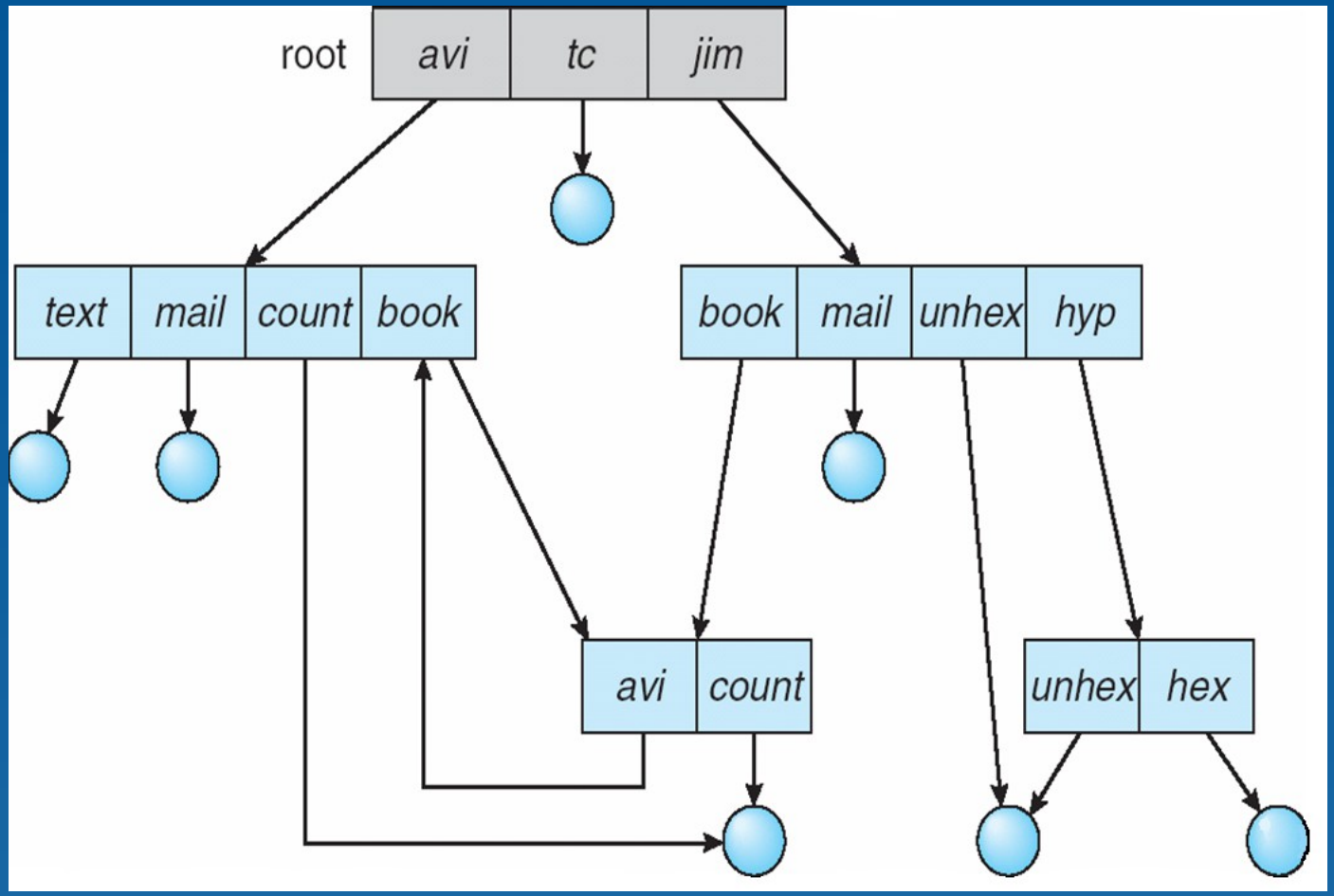
- 能够共享子目录、文件



有向无环图目录结构（续）

- 文件名可以使用别名 (aliasing)
- *dict* 下删除了 w 指向的 *list* ⇒ words 下的指针 “悬空”
- 解决方案
  - 加 “逆向指针”，在它指示下删除指向 list 的指针
    新问题：“逆向指针” 总数不确定
  - 加 “引用计数器”
- 新的 “目录项” 类型
  - **链接 (Link)** – 指向已有文件的链接（指针）
  - **Resolve the link** – 顺着指针定位文件

图目录结构


root
avi
tc
jim
text
mail
count
book
book
mail
unhex
hyp
avi
count
unhex
hex

图目录结构（续）

- 存在环，怎么办？
  - 规定 Links 只能指向文件，不能指向子目录
  - Garbage collection 算法清理孤立环
  - 每当创建 link，调用环检测算法，确保不存在环


End