

文件

常考知识点

标黄为重点

<u>UNIX相关知识</u>	<div>1. UNIX采用树型目录结构，文件信息存放在索引节点中，超级块是用来描述文件系统的</div> <div>2. UNIX系统所有设备都被视为特殊的文件</div> <div>3. UNIX每个目录文件中，默认有两个目录项，“.”表示当前目录，“..”表示父目录</div>
<u>各种表、图</u>	<div>1. <u>打开文件表</u>：存放已打开文件信息的表，将指名文件的属性从外存复制到内存，再使用该文件时直接返回索引</div> <div>2. <u>位图</u>：详见磁盘管理方法</div> <div>3. <u>空闲盘链表</u>：详见磁盘管理方法</div> <div>4. <u>索引表</u>：记录每个文件所存放的盘块地址</div> <div>5. <u>系统调用表</u>：一张由指向实现各种系统调用的内核函数的函数指针组成的表，该表可以基于系统调用函数进行索引，来定位函数地址，完成系统调用</div>
<u>文件性能相关</u>	<div>• <u>与单个文件长度/大小有关的因素</u>：文件块大小，地址项个数，间接地址索引的级数（索引节点总数与单个文件大小无关）</div> <div>• <u>提高文件访问速度的方法</u>：提前读，延迟写，为文件分配连续的簇，采用磁盘高速缓存</div> <div>• <u>文件在磁带上采用连续存放方式</u>；在硬盘上不采用连续存放方法；在内存上采用随机存放方法</div> <div>• <u>连续结构的文件数据读最快</u>，链式的方法不能随机存取，索引的方法占面积</div>
<u>文件打开相关</u>	<div>• <u>文件打开open是把FCB读到内存</u>，不是把文件内容读到内存（即不会读入文件数据，只有用read的时候才会读数据）</div> <div>• <u>open参数包含文件名</u></div> <div>• <u>read/write参数不包含文件名</u>，而是open返回的文件索引</div>
<u>文件保护相关</u>	<div>• <u>防止文件受损的方法→备份</u>：</div> <div>• <u>用于多用户之间的存取权限保护→存取控制矩阵方法</u></div>
<u>其他</u>	<div>• 系统级安全管理包括注册和登陆</div> <div>• 用户确定文件的逻辑结构，由操作系统设计者根据文件存储器的特性确定文件的物理结构，一旦确定，由操作系统管理</div>

基本概念

<u>文件是什么？</u>	<div>• <u>文件是以硬盘为载体的存储在计算机上的信息集合</u></div> <div>• <u>文件可以是文本文档，图片，程序</u></div> <div>• 用户进行的输入输出中，以文件为基本单位</div>
<u>文件由什么组成？</u>	<div>1. <u>一块存储空间</u></div> <div>2. <u>分类和索引的信息</u></div> <div>3. <u>关于访问权限的信息</u></div>
<u>文件有什么特性？</u>	<div>1. <u>可以长期存储在硬盘中</u></div> <div>2. <u>允许可控制的进程间共享访问</u></div> <div>3. <u>能够被组织成复杂的结构</u></div>
<u>文件的属性/元数据</u>	<div>• <u>文件名</u>；<u>文件类型</u></div> <div>• <u>创建者</u>；<u>所有者</u></div> <div>• <u>位置</u>；<u>大小</u>；<u>保护</u>；<u>创建信息</u></div>

文件的数据结构

目录项	相关概念		<ul style="list-style-type: none">• <u>目录项/FCB</u> = 用来记录文件的名字，索引节点指针以及其他目录项的层级关联关系• <u>目录/目录文件</u> = FCB的集合• 为了实现“按名存取”，在文件系统中为每个文件设置用于描述和控制文件的数据结构，称作文件控制块FCB，也叫做目录项• 目录也被视作一个文件，该文件叫做目录文件• 目录项是由内核维护的一个数据结构，缓存在内存，目录项文件存储在磁盘• 目录文件存放该目录中所有子目录文件和数据文件的目录
	包含信息		<ul style="list-style-type: none">• <u>基本信息</u>：文件名；文件物理位置，文件逻辑结构，文件物理结构• <u>存取控制信息</u>：文件主或核准用户或一般用户的存取权限• <u>使用信息</u>：文件创立时间，上次修改时间
索引节点	概念		<ul style="list-style-type: none">• <u>索引节点</u>是用来记录文件的元信息，是文件的唯一表示• 索引节点同样占用磁盘空间• 将文件描述信息从目录项中分离，即应用了索引节点的方法• 使用索引节点可以减少查找文件时的I/O信息量
	分类	磁盘索引节点	<ul style="list-style-type: none">• 指存放在磁盘上的索引节点，每个文件有一个唯一的磁盘索引节点• <u>包含内容</u>：文件主标识符；文件类型；文件存取权限；文件物理地址；文件长度；文件链接计数；文件存取时间
		内存索引节点	<ul style="list-style-type: none">• 指存放在内存中的索引节点，文件打开后，将磁盘索引节点复制到内存中• <u>新增内容</u>：索引节点编号；状态；访问计数；逻辑设备号；链接指针
示例图	书的目录举例		计算机的目录举例
	<div>目录</div> <div>第1章 目录项入门从属关系.....1</div> <div>1.1 UNIX、Linux 和 GNU 简介.....1</div> <div>1.1.1 什么是 UNIX 目录项.....1</div> <div>1.1.2 什么是 Linux.....2</div> <div>1.1.3 GNU 项目和自由软件基金会.....3</div> <div>1.1.4 Linux 发行版.....3</div> <div>1.2 Linux 程序设计.....4</div> <div>1.2.1 Linux 程序.....4</div> <div>1.2.2 文本编辑器.....5</div> <div>1.2.3 C 语言编译器.....5</div> <div>1.2.4 开发系统导引.....7</div> <div>1.3 获得帮助.....12</div> <div>1.4 小结.....14</div> <div>第2章 shell 程序设计.....15</div> <div>2.1 为什么使用 shell 编程.....15</div> <div>2.2 一点哲学.....16</div> <div>2.3 什么是 shell.....16</div> <div>2.4 管道和重定向.....18</div> <div>2.4.1 重定向输出.....18</div> <div>2.4.2 重定向输入.....19</div> <div>2.4.3 管道.....19</div> <div>对应的页码即计算机中的“索引结点”</div>		<div>目录：目录项的集合</div> <div>目录项</div> <div>父目录</div> <div>file1</div> <div>子目录/文件列表 (file1, file2)</div> <div>/etc</div> <div>索引节点指针</div> <div>父目录</div> <div>file1</div> <div>索引节点指针</div> <div>这部分在磁盘上</div> <div>磁盘</div> <div>超级块</div> <div>索引节点区</div> <div>索引节点1</div> <div>索引节点2</div> <div>数据块区</div> <div>逻辑块2</div> <div>逻辑块1</div>

文件的操作			
基本操作	• 创建文件；写文件；读文件；重新定位文件；删除文件；截断文件		
文件打开的过程	<p>• 文件打开就是调用open，根据文件名搜索目录，将指明文件的属性（包括该文件在外上的物理地址）</p> <p>从外存复制到内存打开文件表的一个表目中，并将该表目的编号（也叫索引）返回给用户</p> <p>• 【打开文件操作的主要工作就是把指定文件的目录复制到内存指定的区域】</p> <p>• 【open操作会把文件的FCB调入内存，而不会把文件内容读到内存，只有进程希望获取文件内容时才会读入文件内容】</p>		
文件打开和关闭关联的信息	<p>• 文件指针；文件打开计数；文件磁盘位置；访问权限</p> <p>文件描述符是打开文件的标识</p>		
写入文件的过程	例图	<pre>graph LR; subgraph "用户空间"; W[write()]; end; subgraph "虚拟文件系统"; SW[sys_write()]; end; subgraph "文件系统"; FSW[文件系统的写方法]; end; subgraph "磁盘"; D[(磁盘)]; end; W --> SW; SW --> FSW; FSW --> D;</pre>	
	代码	<pre>fd = open(name, flag);# 打开文件 write(fd,...);# 写数据 read(fd,...);# 读数据 close(fd);# 关闭文件</pre>	
	解释	<p>• 首先用 open 系统调用打开文件，open 的参数中包含文件的路径名和文件名</p> <p>• 使用 write 写数据，其中 write 使用 open 所返回的文件描述符，并不使用文件名作为参数【read同理，结合C语言文件读写代码记忆】</p> <p>• 使用完文件后，要用 close 系统调用关闭文件，避免资源的泄露</p>	

文件的保护			
1. 保护的 <u>目的</u> ：解决对文件的读，写，执行的许可问题			
2. 一个文件的访问常由 <u>用户访问权限</u> 和 <u>文件属性</u> （包括保存在FCB中对文件访问的控制信息）共同设置			
3. 保护的 <u>方式</u>			
非访问控制方法		1. 口令	2. 加密保护
	定义	<ul style="list-style-type: none">• 用户建立一个文件时需要提供口令• 用户请求访问时必须提供相应口令	<ul style="list-style-type: none">• 对文件进行加密，访问时需要密钥
	优点	<ul style="list-style-type: none">• 时间空间开销不多	<ul style="list-style-type: none">• 保密性强，节省了存储空间
	缺点	<ul style="list-style-type: none">• 口令直接存在系统内部，不安全	<ul style="list-style-type: none">• 编码和译码需要时间
访问控制方法	<ul style="list-style-type: none">• <u>访问控制的目的</u>：用于控制用户对文件的访问方式• <u>访问控制的对象</u>：读；写；执行；添加；删除；列表清单• 访问控制机制必须由系统实现		
		方法一	方法二
	定义	<ul style="list-style-type: none">• 为每个文件和目录增加一个访问控制列表ACL• 该表规定每个用户名及其所允许的空间管理	<ul style="list-style-type: none">• 采用精简的访问列表• 该列表采用<u>拥有者</u>，<u>组</u>和<u>其他</u>三种用户类型
	优点	<ul style="list-style-type: none">• 可以使用复杂的访问方法	<ul style="list-style-type: none">• 只需要三个域即可列出访问表中这三类用户的访问权限
	缺点	<ul style="list-style-type: none">• 长度无法预计并且可能导致复杂的空间管理	

文件的逻辑结构【用户角度的文件组织形式】

定义	<ul style="list-style-type: none">即文件中的数据在逻辑层面是如何组织起来的	
无结构文件/ 流式文件	<ul style="list-style-type: none">最简单的文件组织形式，是有序相关信息项的集合，以字节为单位对基本信息单元操作不多的文件适合该方式，如源代码文件，目标代码文件	
有结构文件/ 记录式文件	1. 顺序文件	<ul style="list-style-type: none">串结构：只能按顺序查找，费时顺序结构：可采用折半查找，检索效率高顺序查找平均次数$\frac{N}{2}$；索引顺序查找平均次数$N^{\frac{1}{2}}$
	2. 索引文件	<ul style="list-style-type: none">提高了存取速度，但索引表增加了存储空间
	3. 索引顺序文件	<ul style="list-style-type: none">提同一组中的关键字可以无序，但组与组之间的关键字必须有序先通过索引表查找所在的组，然后再该组中使用顺序查找

文件的物理结构【文件在外存上的存储组织形式】

定义	<ul style="list-style-type: none">研究文件数据在物理存储设备上是如何分布和组织的文件在磁带上→连续存放方式文件在磁盘上→不采用连续存放方式文件在内存上→随机存放方式																
文件的存储空间管理	<ul style="list-style-type: none">文件的存储空间管理就是对磁盘空闲块的管理Linux 文件系统就采用了位图的方式来管理空闲空间，不仅用于数据空闲块的管理，还用于 inode 空闲块的管理，因为 inode 也是存储在磁盘的																
	空闲表法	<div><div><ul style="list-style-type: none">为所有空闲空间建立一张表表内容包括空闲区的第一个块号和该空闲区的块个数</div><table><tr><th>序号</th><th>第一个空闲块号</th><th>空闲块个数</th></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>3</td><td>13</td><td>2</td></tr><tr><td>4</td><td>...</td><td>...</td></tr></table><div>磁盘块 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</div></div>	序号	第一个空闲块号	空闲块个数	1	2	3	2	8	3	3	13	2	4
	序号	第一个空闲块号	空闲块个数														
	1	2	3														
	2	8	3														
3	13	2															
4															
空闲链表法	<div><div><ul style="list-style-type: none">每一个空闲块里有一个指针指向下一个空闲块这样能很方便的找到空闲块并管理起来</div><div>通过链表的方式将空闲块组织起来</div><div>磁盘块</div><div>空闲块 已分配的块</div></div>																
位图法	<div><div><ul style="list-style-type: none">位图是利用二进制的一位来表示磁盘中一个盘块的使用情况磁盘上所有的盘块都有一个二进制位与之对应</div><div><ul style="list-style-type: none">当值为 0 时，表示对应的盘块空闲值为 1 时，表示对应的盘块已分配通常可用$m*n$个数来构成m行n列的位示图且$m*n$等于磁盘的总块数</div></div>																
成组链表法	<ul style="list-style-type: none">结合空闲表和空闲链表的优点，克服表长的缺点																

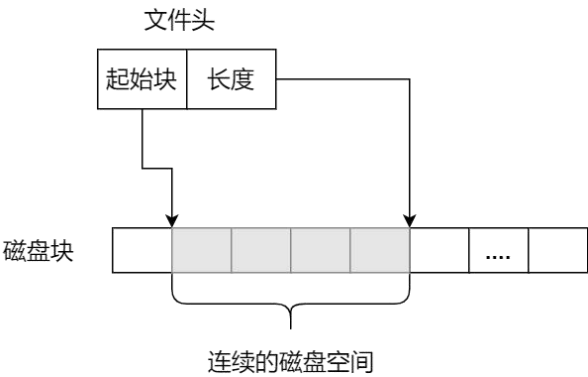
文件的物理结构【文件在外存上的存储组织形式】

• 文件的存储就是对磁盘非空闲块的管理

方式	访问磁盘次数	优点	缺点
顺序分配	需访问磁盘1次	顺序存取速度快，当文件是定长时可以根据文件起始地址及记录长度进行随机访问	要求连续的存储空间，会产生外部碎片，不利于文件的动态扩充
链表分配	需访问磁盘n次	无外部碎片，提高了外存空间的利用率，动态增长比较方便	智能按照文件的指针链顺序访问，查找效率低，指针信息存放消耗内存或磁盘空间
索引分配	m 级需访问磁盘 m+1 次	可以随机访问，易于文件的增删	索引表增加存储空间的开销，索引表的查找策略对文件系统效率影响较大

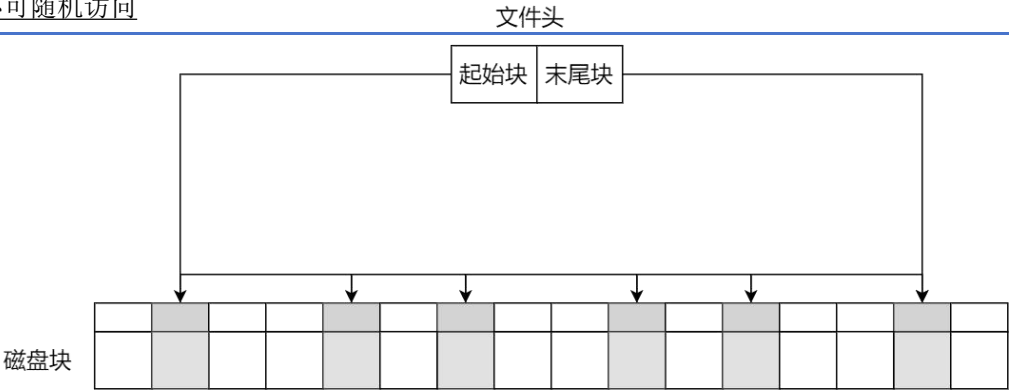
1. 顺序分配【类似数组】

- 文件头需指定起始块的位置和长度
- 要求连续的存储空间
- 可随机访问



2. 链表分配【类似链表】

- 存放是离散的，不用连续的，于是就可以消除磁盘碎片
- 可大大提高磁盘空间的利用率，同时文件的长度可以动态扩展
- 隐式链接无法直接访问数据块，只能通过指针顺序访问文件
- 显式链接把用于链接文件各数据块的指针，显式地存放在内存的一张链接表中
- 内存中的这样一个表格称为文件分配表FAT (File Allocation Table)
- 不可随机访问

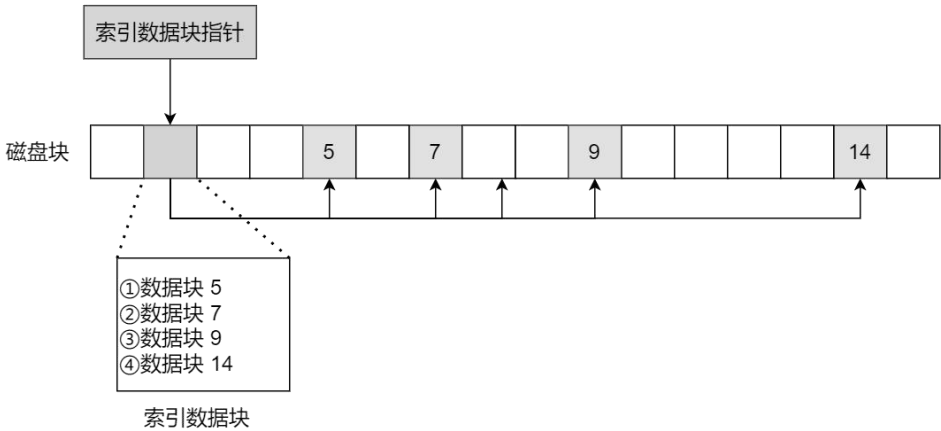


文件的
存储方式

文件的物理结构【文件在外存上的存储组织形式】

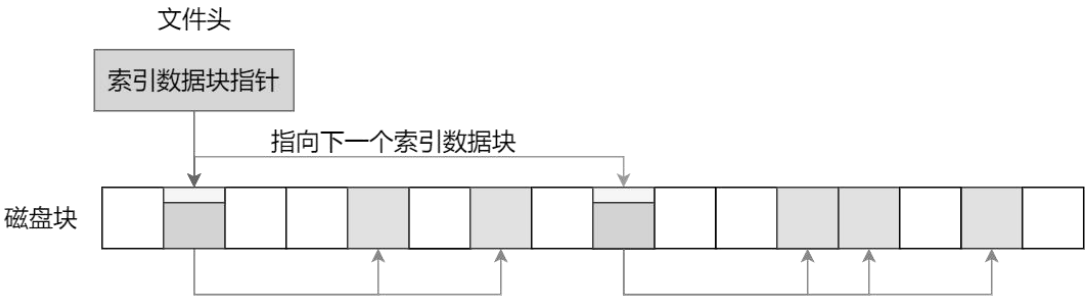
3. 索引分配【类似索引表】

- 为每个文件创建一个「索引数据块」
- 里面存放的是指向文件数据块的指针列表
- 像书的目录一样，要找哪个章节的内容，看目录查就可以
- 可随机访问



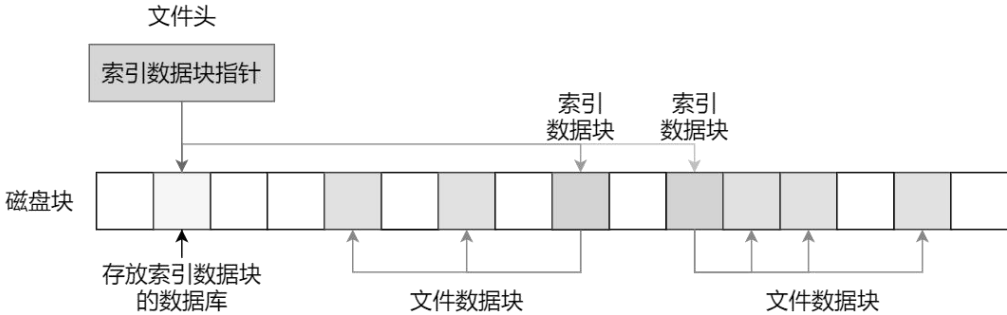
4. 链表 + 索引

- 索引数据块留出一个存放下一个索引数据块的指针



5. 索引+索引【多级索引】

- 通过一个索引块来存放多个索引数据块



多级索引举例→unix系统的inde结构

- 对于小文件使用直接查找的方式可减少索引数据块的开销
- 对于大文件则以多级索引的方式来支持
- 所以大文件在访问数据块时需要大量查询
- 存放文件所需的数据块小于 10 块，则采用直接查找的方式
- 存放文件所需的数据块超过 10 块，则采用一级间接索引方式
- 前面两种方式都不够存放大文件，则采用二级间接索引方式
- 二级间接索引也不够存放大文件，则采用三级间接索引方式

文件的
存储方式

文件系统

基本概念和目标

概念	<ul style="list-style-type: none">文件系统 = OS中负责管理持久数据的子系统文件系统 = 与文件管理有关的软件 + 被管理的文件 + 试试文件管理所需的数据结构文件系统需先挂在到某个目录才可正常使用文件的基本操作单位就是数据块	
目标	<ol style="list-style-type: none">实现对文件的基本操作 = 按名存储和查找文件 + 组织成合适的结构 + 文件共享 + 文件保护【用户角度】管理与磁盘的信息交换 + 完成逻辑结构和物理结构的变换【OS角度】组织文件在磁盘上的存放 + 采取好的文件排放顺序和磁盘调度方法【OS角度】	
分类	磁盘的文件系统	<ul style="list-style-type: none">它是直接把数据存储在磁盘中，比如 Ext 2/3/4、XFS 等都是这类文件系统
	内存的文件系统	<ul style="list-style-type: none">这类文件系统的数据不是存储在硬盘的，而是占用内存空间我们经常用到的 /proc 和 /sys 文件系统都属于这一类读写这类文件，实际上是读写内核中相关的数据
	网络的文件系统	<ul style="list-style-type: none">用来访问其他计算机主机数据的文件系统，比如 NFS、SMB 等等

文件系统的层次结构

	主要功能和介绍	
I/O控制	设备驱动程序	<ul style="list-style-type: none">将输入的命令翻译成底层硬件的特定指令
	中断处理程序	<ul style="list-style-type: none">利用指令使IO设备与系统交互
基本文件系统	<ul style="list-style-type: none">向对应的设备驱动程序发送通用命令，以读取和写入磁盘的物理块管理内存缓冲区，保存各种文件系统，目录和数据块的缓冲	
文件组织模块	<ul style="list-style-type: none">组织文件及其逻辑块和物理块可以将逻辑地址转换为物理地址有空闲空间管理器，以跟踪未分配的块，根据需要提供给文件组织模块	
逻辑文件系统	<ul style="list-style-type: none">用于管理元数据信息（包括文件系统的所有结构，不包括文件内容）管理目录结构通过FCB维护文件结构负责文件保护	

虚拟文件系统VFS

目的	<ul style="list-style-type: none">为用户程序提供了文件系统操作的统一接口，屏蔽了不同文件系统差异和操作细节
特性	<ol style="list-style-type: none">能提高系统性能不是一种实际的文件系统只存在于内存中，不存在与任何外存空间中在系统启动时建立，在系统关闭时消亡
VFS的数据结构	<ol style="list-style-type: none">超级块对象引节点对象目录项对象文件对象

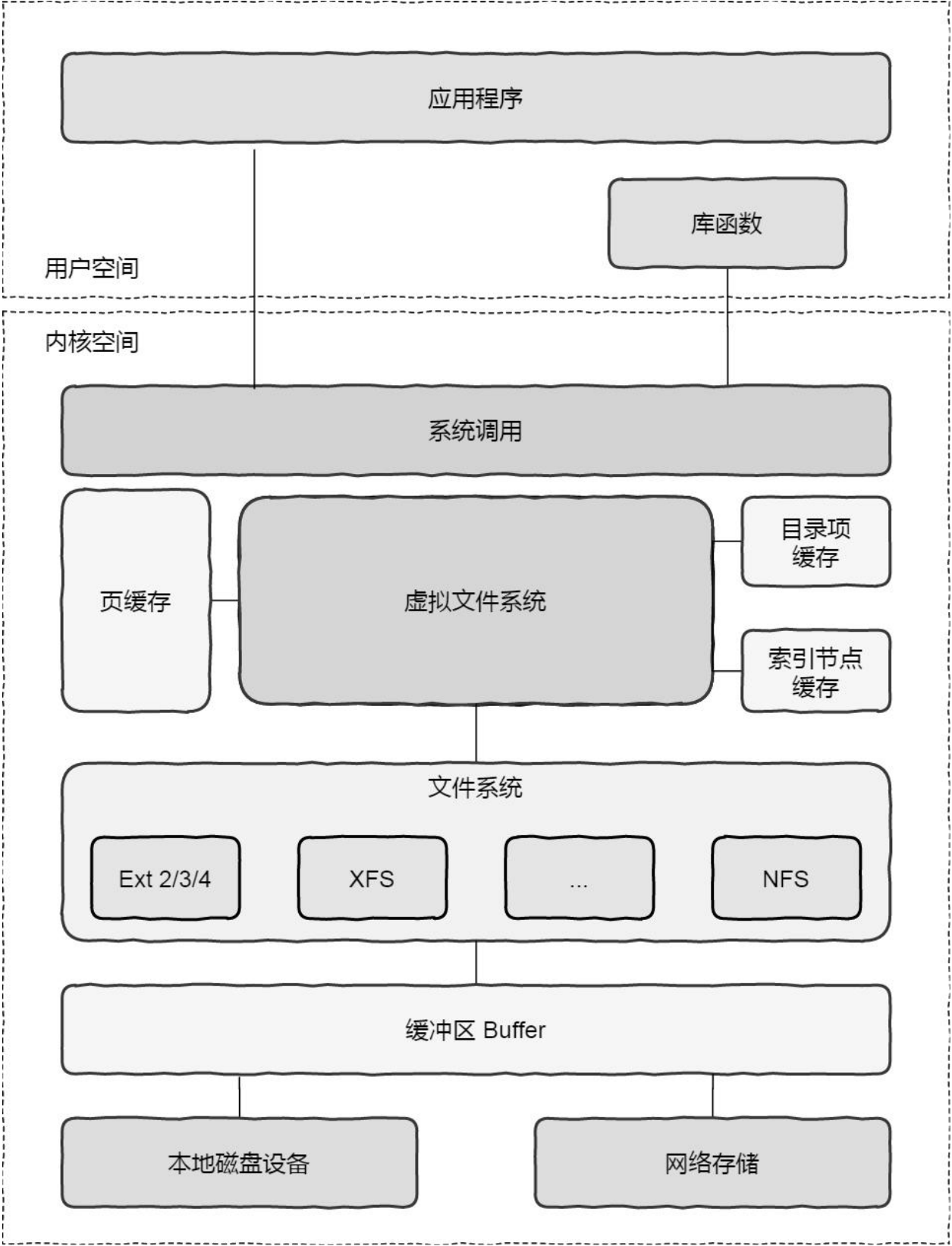
文件系统的布局

在磁盘中的结构	<div>1 K</div> <div>引导块 块组 0 块组1 ... 块组n</div> <div><div>.....</div></div>		
	超级块	块组描述符表	
	数据位图	inode 位图	
	inode 列表	数据块	
	1 块	多块	
	1 块	1 块	
	多块	多块	
<ul style="list-style-type: none">• 最前面的第一个块是引导块，在系统启动时用于启用引导• 接着后面就是一个一个连续的块组了，块组的内容如下			
超级块	<ul style="list-style-type: none">• 包含的是文件系统的重要信息• 比如 inode 总个数、块总个数、每个块组的 inode 个数、每个块组的块个数		
块组描述符	<ul style="list-style-type: none">• 包含文件系统中各个块组的状态• 比如块组中空闲块和 inode 的数目等，每个块组都包含了文件系统中「所有块组的组描述符信息」		
数据位图 inode 位图	<ul style="list-style-type: none">• 用于表示对应的数据块或 inode 是空闲的，还是被使用中		
inode 列表	<ul style="list-style-type: none">• 包含了块组中所有的 inode，inode 用于保存文件系统中与各个文件和目录相关的所有元数据		
数据块	<ul style="list-style-type: none">• 包含文件的有用数据		
在内存中的结构	<ul style="list-style-type: none">• 内存中的信息用于管理文件系统并通过缓存来提高信息• 这些结构有以下类型：<ol style="list-style-type: none">1. 内存中的安装表2. 内存中的目录结构的缓存包含最近访问目录的信息3. 整个系统的打开文件表4. 每个进程的打开文件表		

分区和安装

- 一个磁盘可划分为多个区，每个分区都可以创建单独的文件系统，每个分区都可包含不同的操作系统
- 文件在使用前必须先安装（即挂载）

用户空间，系统调用，虚拟文件系统，缓存，文件系统和存储之间的关系



目录

常考点

目录结构相关	1. 在树型目录中，为了加快文件检索速度，可设置当前目录，于是文件路径可以从当前目录开始查找 2. 文件系统采用多级目录的目的是解决命令冲突 3. 树形目录结构中，用户对文件的首次访问通常采用文件路径名，之后对文件的访问通常使用文件描述符
目录检索相关	• 在顺序检索法时，只要路径名的一个分量名未找到，就应停止查找
文件共享相关	• 用户进程的打开文件表关于同一个文件不一定相同，例如读写指针位置不一定相同
其他	• 整个文件系统只有一个系统文件打开表（里面的表项都是不重复的），同一文件打开多次只需改变引用计数

目录管理要求

1. 实现“按名存取”
 2. 要提高目录的检索速度
 3. 需要提供用于控制访问文件的信息
 4. 允许不同用户对不同文件采用系统的名字

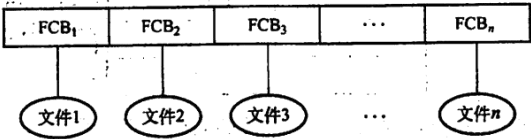
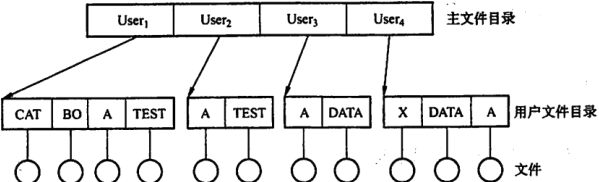
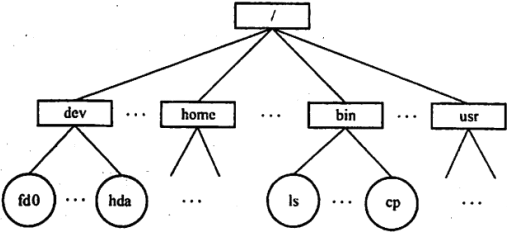
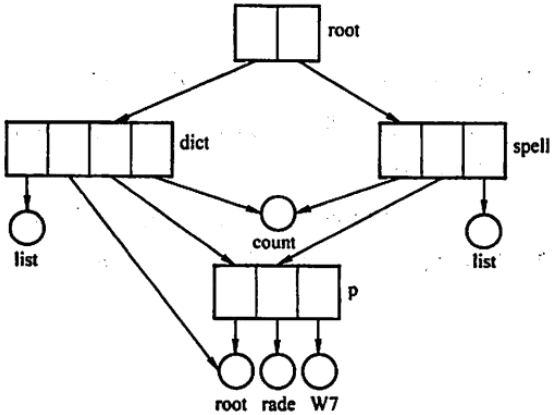
目录的操作

搜索文件 修改目录 创建文件 创建目录 删除目录 移动目录 显示目录	删除文件：删除一个文件时，会根据FCB回收相应的磁盘空间，将FCB回收，并删除目录中对应的目录项
--	--

目录的查询/检索

概念	• 目录查询通过在磁盘上反复搜索完成，需要不断进行I/O操作，开销大 • 可以把当前使用的文件目录复制到内存，从而降低磁盘操作次数，提高系统速度			
实现方法		定义	优点	缺点
	线性列表 【对应线性查找】	采取线性列表存储文件目录项	实现简单	查找费时
	哈希表 【对应散列查找】	采取哈希表存储文件目录项	查找迅速，插入删除简单	需要一些措施来避免冲突

目录结构

	单级目录结构	两级目录结构
定义	<ul style="list-style-type: none">整个文件系统只建立一张目录表每个文件占一个目录项	<ul style="list-style-type: none">文件目录分为主文件目录MDF和用户文件目录UFDMDF记录用户名UFD所在的存储位置UFD记录用户文件的FCB信息
优点		<ul style="list-style-type: none">解决了多用户之间的文件重名问题文件系统可以在目录上实现访问限制
缺点	<ul style="list-style-type: none">查找速度慢文件不允许重名不便于文件共享不适合多用户的OS	<ul style="list-style-type: none">缺乏灵活性，不能对文件分类
结构图	 <p>图 4.11 单级目录结构</p>	 <p>图 4.12 两级目录结构</p>
	树形目录结构	无环图目录结构
定义	<ul style="list-style-type: none">使用绝对路径，相对路径，当前路径的结构不同的用户的文件，文件名可相同可不同大多OS采用这种目录结构	<ul style="list-style-type: none">在树形目录结构上加入有向边，组成一个有向无环图
优点	<ul style="list-style-type: none">可以很方便的对文件进行分类能够有效地进行文件的管理和保护	<ul style="list-style-type: none">实现了文件共享
缺点	<ul style="list-style-type: none">利于文件共享查找文件增加了磁盘访问次数，会影响查询速度	<ul style="list-style-type: none">使系统的管理变得更加复杂
结构图	 <p>图 4.13 树形目录结构</p>	 <p>图 4.14 无环图目录结构</p>

文件共享

概念	• 文件共享使多个用户共享同一个文件，系统只需保留该文件的一个副本																	
文件共享方式		基于索引节点的关系方式【硬链接】	基于符号链实现文件共享【软链接】															
	定义	<ul style="list-style-type: none">硬链接就是多个指针指向一个索引节点文件的物理地址和其他文件属性信息放在索引节点中	<ul style="list-style-type: none">软链接相当于重新创建一个文件新文件只包含被链接文件的路径名															
	特点	<ul style="list-style-type: none">硬链接不可用于跨文件系统硬链接查找速度比软链接快	<ul style="list-style-type: none">软链接可以跨文件系统															
	新增文件时	<ul style="list-style-type: none">建立硬链接时，引用计数值加1	<ul style="list-style-type: none">符号链接，计数值直接复制															
	删除文件时	<ul style="list-style-type: none">删除文件时，计数值减1若得到的值不为0，则不能删除此文件即只要还有一个指针在，索引节点就不会被删除	<ul style="list-style-type: none">删除操作对符号链接不可见以后再通过符号链接访问时，若发现文件不存在，直接删除符号链接															
	示例	<div><div>Wang用户文件目录</div><table><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td>Test r</td><td></td></tr><tr><td></td><td></td></tr></table><div>Lee用户文件目录</div><table><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td>Test r</td><td></td></tr><tr><td></td><td></td></tr></table><div>索引节点</div><div>count=2 文件物理地址</div><div>Test</div></div> <td>快捷方式属于文件共享中的软链接</td>					Test r								Test r			
Test r																		
Test r																		

图 4.15 基于索引结点的共享方式