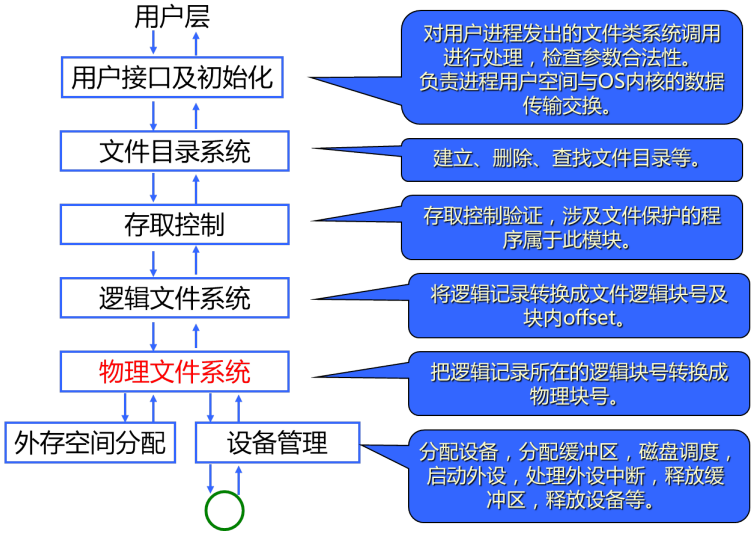

河南大学
操作系统
计算机学院

第 8 章

磁盘存储器的管理

文件系统的基本模型


用户层
用户接口及初始化
文件目录系统
存取控制
逻辑文件系统
物理文件系统
外存空间分配
设备管理
对用户进程发出的文件类系统调用进行处理，检查参数合法性。负责进程用户空间与OS内核的数据传输交换。
建立、删除、查找文件目录等。
存取控制验证，涉及文件保护的程序属于此模块。
将逻辑记录转换成文件逻辑块号及块内offset。
把逻辑记录所在的逻辑块号转换成物理块号。
分配设备，分配缓冲区，磁盘调度，启动外设，处理外设中断，释放缓冲区，释放设备等。

文件的物理结构（文件的存储结构）

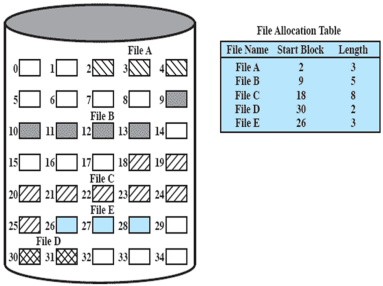
外存的组织方式

1. 连续组织方式（顺序分配）
2. 链接组织方式
3. 索引组织方式

连续组织方式

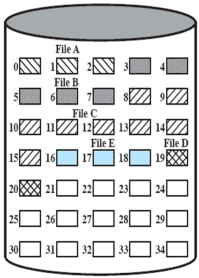
- 连续组织方式为每一个文件分配一片连续的磁盘块。例如，第一个盘块的地址为 b，则第二个盘块的地址为 b+1，第三个盘块的地址为 b+2。通常它们都位于一条磁道上，在进行读/写时不必移动磁头，仅当访问到磁道的最后一个盘块后才需要移到下一条磁道。
- 这样所形成的文件结构称为顺序文件结构，此时的物理文件称为顺序文件。

连续组织方式



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

- 文件目录项中只需要保存文件的起始块号和长度（连续的块数）。
- 连续组织方式会形成外部碎片。可以通过紧缩 (compact) 技术把外存空闲空间合并成连续的区域。

连续组织方式

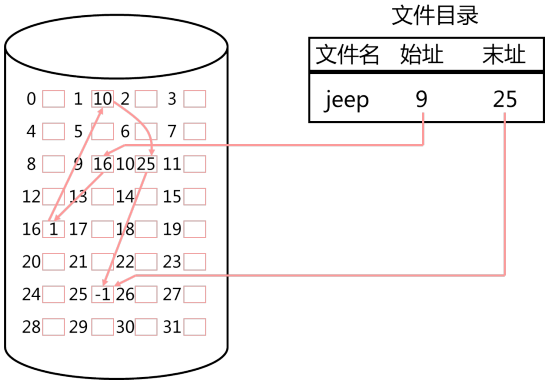
- 优点
  - 顺序访问比较容易。
  - 访问速度快，磁头移动距离少（换磁道时才移动）。
  - 保证了逻辑文件中的记录顺序与存储器中文件占用盘块的顺序的一致性。从逻辑地址映射到物理地址较简单。
- 缺点
  - 要求有连续的存储空间。
  - 存在外部碎片，浪费空间。如果定期紧凑来消除碎片，则又需花费大量的机器时间。
  - 必须事先知道文件的长度。有时需要估计（预留空间问题）。

链接组织方式

- 链接组织方式，也称为链接分配，采用离散分配方式，可以消除外部碎片。
- 链接分配中，每个文件有一个盘块的链接列表，把所需分配磁盘块链接在一起。
- 文件目录项中只需要保存文件的起始盘块地址，每个块中有指向下一个块的指针。
- 物理块可以分散在磁盘各处（离散分配）。
- 链接分配分为隐式链接和显式链接两种。

隐式链接

- 采用隐式链接分配方式时，在文件目录的每个目录项中，保存指向链接文件第一个盘块和最后一个盘块的指针。



隐式链接

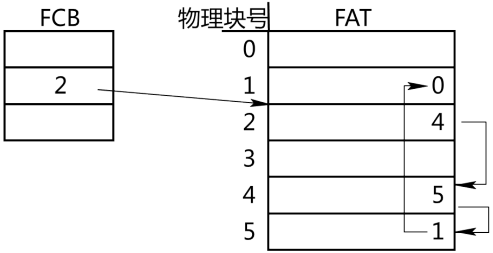
- 隐式链接分配采用离散分配方式，消除了外部碎片。
- 隐式链接分配的特点：
  1. 只适合于顺序访问，对随机访问极其低效。如果要访问文件所在的第 i 个盘块，则必须先读出文件的第一个盘块，···，顺序地查找直至第 i 块（访问第 i 盘块，需要启动 i 次磁盘）。
  2. 可靠性差，任何一个指针出现问题，都会导致整个链的断开。
  3. 指针占存储空间。如果指针占用 4 个字节，对于盘块大小为 512 字节的磁盘，则每个盘块中只有 508 个字节可供用户使用。

显式链接

- 显式链接：把用于链接的指针显式地存放在内存的一张表中，查找在内存中进行。
- 文件分配表 FAT (File Allocation Table)
  - 文件分配表在整个逻辑磁盘分区仅设置一张。
  - 表序号为整个磁盘的物理块号。
  - 表项中存放链接指针，即下一个块号。
  - 文件的首块号存入相应文件的 FCB 中。
- 特点：FAT 表在内存中，查找记录的过程是在内存中进行的，不仅显著地提高了检索速度，而且大大减少了访问磁盘的次数。

显式链接

- 文件检索过程：文件控制块 FCB ⇒ FAT ⇒ 块链



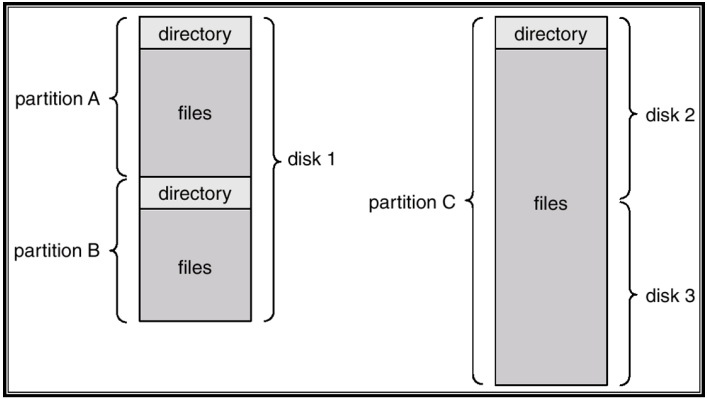
链接组织方式的优缺点

- 优点
  - 无外部碎片，没有磁盘空间浪费，不需紧缩磁盘空间。
  - 无需事先知道文件大小。文件动态增长时可以动态分配空间，对文件的增、删、改十分方便。
- 缺点
  - 隐式链接分配：不支持高效随机/直接访问，仅适合于顺序存取。查找某一个块必须从第一块开始沿指针进行。
  - 隐式链接分配：需为指针分配空间。
  - 隐式链接分配：可靠性较低（指针丢失/出错）。
  - 显式链接分配：FAT 表需占用较大的内存空间。

FAT 文件系统（链接组织方式）

- FAT 文件系统是由微软发明并带有部分专利的文件系统。FAT：File Allocation Table 文件分配表。
- FAT 文件系统支持把一个物理磁盘分成 4 个逻辑磁盘（卷、分区），一个卷也可以由多个物理磁盘组成（RAID）。
- 在每个分区中都配有两张相同的文件分配表 FAT1 和 FAT2，在 FAT 的每个表项中存放下一个盘块号，它实际上是用于盘块之间的链接的指针，通过它可以将一个文件的所有的盘块链接起来。
- FAT 文件系统共三个版本：FAT12，FAT16 和 FAT32。

磁盘布局（分区）



多个分区可共享一个磁盘
一个分区可以跨多个磁盘

簇（cluster）的概念

- FAT 文件系统引入了一个新的分配单位：簇
  - 若干个连续的扇区 (sector) 或盘块称为一簇，簇的大小一般是 2^n 个扇区或盘块。
  - FAT 文件系统把簇作为一个虚拟扇区。文件的存储空间通常由多个簇组成。
- 如何选择簇的大小
  - 两个极端：大到能容纳整个文件，小到一个扇区。
  - 如果簇较大：可提高 I/O 访问性能，减少访问时间，但簇内碎片浪费问题较严重。
  - 如果簇较小：簇内的碎片浪费较小，但可能存在簇编号空间不够的问题（如 FAT12、16）。

FAT12 文件系统

- FAT12 文件系统：每个 FAT 表的表项数占 12 位，即 FAT 表最多允许 4096 个表项。
- 如果采用盘块作为基本分配单位，每个盘块（扇区）的大小一般是 512 字节，则每个磁盘分区的容量为 2MB（4096×512 B）。一个物理磁盘最多支持 4 个逻辑磁盘分区，所以可管理的磁盘最大容量为 8MB。
- 磁盘的容量超过了 8 MB，FAT12 是否还可继续用呢？

FAT12 文件系统

- FAT12 文件系统：每个 FAT 表的表项数占 12 位，即 FAT 表最多允许 4096 个表项。
- 如果采用盘块作为基本分配单位，每个盘块（扇区）的大小一般是 512 字节，则每个磁盘分区的容量为 2MB（4096×512 B）。一个物理磁盘最多支持 4 个逻辑磁盘分区，所以可管理的磁盘最大容量为 8MB。
- 磁盘的容量超过了 8 MB，FAT12 是否还可继续用呢？
- 如果采用簇（cluster）作为基本分配单位，当一个簇包含 8 个扇区时，磁盘容量可以达到 64MB。

FAT12 文件系统

- 磁盘分区容量与簇大小的关系
  - 如果磁盘分区容量增加，而簇的总数保持不变即簇编号所需位数保持不变，则簇就变大。缺点是簇内碎片浪费增加。
  - 如果磁盘分区容量增加，而簇的大小不变，则簇的总数就会增加，相应地簇的编号所需要的位数增加。
- 以簇作为基本分配单位的优点：
  1. 可以管理较大容量磁盘。
  2. 在相同的磁盘容量条件下，可以减少 FAT 表的表项数，使 FAT 表占用更少的存储空间，减少了访问 FAT 表的存取开销，提高了文件系统的效率。

FAT16 文件系统

- FAT16 文件系统：每个 FAT 表的表项数占 16 位，即 FAT 表最多允许 65536 个表项。
- 一个磁盘分区可以管理 65536(2^{16}) 个簇。在 FAT16 的每个簇中可以有的盘块数为 4、8、16、32、64。如果每个簇中最大的盘块或扇区数为 64，FAT16 可以管理的最大分区空间为 $2^{16} \times 64 \times 512$ = 2048 MB。（不引入簇为 32MB）
- 优点：可以管理更大的容量磁盘。
- 缺点：当磁盘容量迅速增加时，FAT16 文件系统簇内碎片所造成的浪费比较严重。

FAT32 文件系统

- FAT32 文件系统：每个 FAT 表的表项数占 32 位，即 FAT 表最多允许 2^{32}= 4,294,967,296 个表项。
- FAT32 可以管理更多的簇，这样就允许采用较小的簇（较少簇内碎片），通常 FAT32 的每个簇都固定为 4 KB（如果盘块大小为 512B，则每簇包括 8 个扇区）。
- FAT32 最大支持 2TB 的磁盘，支持的最大磁盘分区为 32GB（但不支持小于 512MB 的分区）。
- 优点：支持更小的簇和更大的磁盘容量。
- 缺点：①FAT 表很大；②单个文件不能大于 4GB（如果拷贝单个文件超过 4G 会显示硬盘空间不足）。

NTFS 文件系统

- NTFS 文件系统：New Technology File System 新技术文件系统，是目前的主流文件系统。
- NTFS 具有很多新特征：
  - NTFS 文件系统使用了 64 位磁盘地址，理论上可以支持 2^{64} 字节的磁盘分区（实际支持最大分区容量为 2TB）。
  - 支持长文件名，单个文件名限制在 255 个字符以内，全路径名为 32767 个字符。
  - 具有系统容错功能。
  - 提供了数据的一致性功能（日志文件和检查点）。
  - 提供了文件加密、文件压缩等功能。

磁盘的组织

- NTFS 也是以簇作为磁盘空间分配和回收的基本单位。一个文件占用若干个簇，一个簇只属于一个文件。
- NTFS 通过簇来间接管理磁盘，不需要知道盘块的大小，使 NTFS 具有了与磁盘物理盘块大小无关的独立性，很容易支持盘块大小不是 512 字节的非标准磁盘。
- 在 NTFS 文件系统中，把卷上簇的大小称为“卷因子”，一个簇包含 2^n(n 为整数) 个盘块。
- 簇的大小可由格式化命令确定或按磁盘容量来确定。为了在传输效率和簇内碎片之间进行折中，NTFS 在大多数情况下簇的大小为 4KB。

磁盘的组织

- NTFS 使用逻辑簇号（LCN）和虚拟簇号（VCN）来进行簇的定位。
- LCN是以卷为单位，把整个卷中所有的簇从头到尾进行简单的编号。
- NTFS 在进行地址映射时，可以通过卷因子与 LCN 的乘积，便可算出卷上的物理字节偏移量，从而得到文件数据所在的物理磁盘地址。
- VCN是以文件为单位，把某个文件的簇按顺序编号。
- 只要知道了文件开始的簇地址，就可以把 VCN 映射到 LCN。

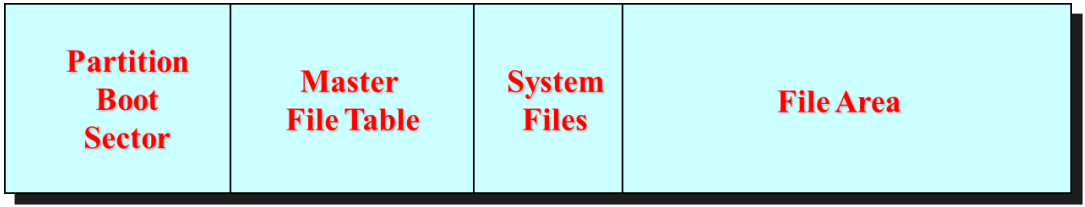
文件的组织

- 在 NTFS 中，以卷为单位，把一个卷中的所有文件信息都以文件记录的方式记录在一张主控文件表 MFT (Master File Table) 中。
- 文件通过主控文件表来确定其在磁盘上的存储位置。
- MFT 为每一个文件保存着一组称为“属性”的记录，在 MFT 中占有一行。每行大小固定为 1 KB，每行称为所对应文件的元数据 (metadata)，也称为文件控制字。
- 当文件较小时，可以把文件的所有属性直接记录在元数据中。
- 当文件较大时，元数据仅记录该文件的一部分属性，文件的 data 属性（文件内容）等记录到卷中的其它簇中。

磁盘布局（FAT）

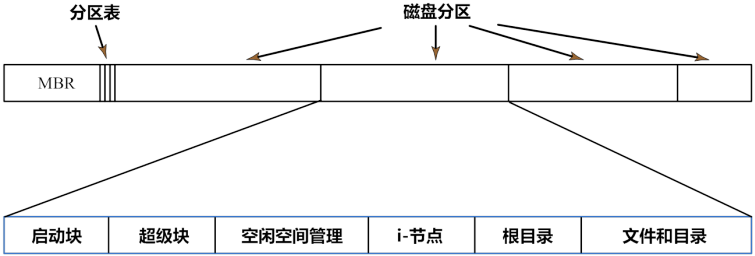
- 文件分配表（FAT）：反映磁盘空间的使用情况，它告诉操作系统，文件存放在磁盘的什么地方。FAT 表共两个，互为备份。
- 根目录区的文件目录表（FDT）：用于登记管理根目录下文件（包括子目录）的名称、类型、文件属性、文件的首簇号以及文件长度等信息的表格。
- FAT32 没有 FDT 表（FAT32 不使用固定的根目录区，根目录与其它子目录一样动态分配空间）

磁盘布局（NTFS）



- Partition Boot Sector：每个分区的引导扇区。
- Master File Table（MTF）：主控文件表，存放本分区中的所有文件（包括目录）的列表及可用磁盘空间。
- System Files（系统文件）：存储日志文件，位示图，属性表等。

磁盘布局（UNIX）



- MBR (Master Boot Record，主引导记录)：计算机引导时，BIOS 从 MBR 扇区（磁盘的 0 扇区）读入并且执行其中的程序。其功能是读入分区表，检查系统的活动分区，读入活动分区的第 1 个扇区。
- 分区表：给出每个分区的起点和终点地址。

磁盘布局（UNIX）

- 启动块(0 号块)：用于系统引导。
- 超级块(1 号块)：包括文件系统的总体信息，比如文件系统的规模、空闲块的数目。
- 空闲空间管理：以位示图形式给出，也可用指针链表形式给出。
- i 节点：每个文件对应一个 i 节点，i 节点包括了除了文件名以外的全部管理信息。
- 根目录：文件系统目录树的顶端。
- 其它部分：除根目录以外的所有子目录和全部文件。

索引组织方式的引入

- 链接分配方式解决了连续分配方式存在的问题（要求连续空间、外部碎片等），但又出现了另外两个问题：
  1. **不能支持高效的直接存取。**隐式链接分配方式查找某一个块必须从第一块开始沿指针进行。显式链接分配方式要对一个较大的文件进行直接存取，必须首先在 FAT 中顺序地查找许多盘块号。
  2. **FAT 需占用较大的内存空间。**由于一个文件所占用盘块的盘块号是随机地分布在 FAT 表中的，因而只有将整个 FAT 表调入内存，才能保证在 FAT 表中找到一个文件的所有盘块号。当磁盘容量较大时，FAT 表可能要占用数兆字节的内存空间，这是令人难以接受的。

索引组织方式

- 事实上在打开某个文件时，只需把该文件占用的盘块编号调入内存即可，没有必要把整个 FAT 表调入内存。
- 改进：把每个文件所对应的盘块号集中地放在一起。
- 索引组织方式就是基于这种思想形成的一种分配方法。
- 索引分配：为每一个文件分配一个索引块（表），再把分配给该文件的所有块号，都记录在该索引块中。索引块就是一个含有许多块号地址的数组。
  - 该索引块的地址由该文件的目录项指出。
  - 支持随机/直接存取。
  - 不会产生外部碎片。
  - 缺点：建立一个文件就需要分配一个索引块，消耗外存空间。适用于文件

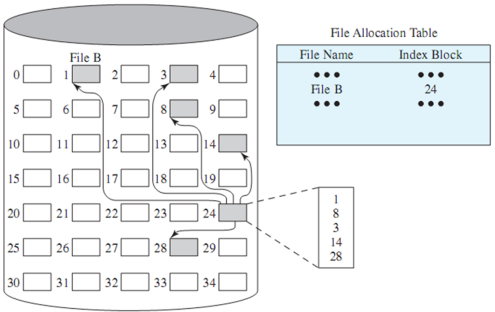
索引组织方式

索引分配

1. 单级索引分配
2. 多级索引分配
3. 混合索引分配

单级索引分配

- 单级索引分配为每个文件分配一个索引块 (表)，把分配给该文件的所有盘块号都记录在该索引块中。
- 在建立一个文件时，只需在该文件的目录项中填上指向索引块的指针。



多级索引分配

- 一个大文件需要的盘块可能很多，其索引块不止一个，需要通过链指针将各索引块按序链接起来。
- 当文件太大，其索引块太多时，这种方法是低效的。
- 多级索引分配为这些索引块再建立一级索引，即系统再分配一个（目录）索引块，作为第一级索引的索引块（两级索引）。
- 如果文件非常大时，还可用三级、四级索引分配方式。

多级索引分配

- 如果每个盘块的大小为 1 KB，每个盘块号占 4 个字节，则在一个索引块中可存放 256 个盘块号。在两级索引时，最多可存放的盘块号总数 N = 256×256 = 64 K 个，即所允许的文件最大长度为 64 MB。
- 如果每个盘块的大小为 4 KB，每个盘块号占 4 个字节，单级索引时所允许的文件最大长度为 1024×4 KB =4MB
- 如果采用两级级索引时最大文件长度为 1024×1024×4 KB =4GB

增量式索引组织方式（混合索引分配）

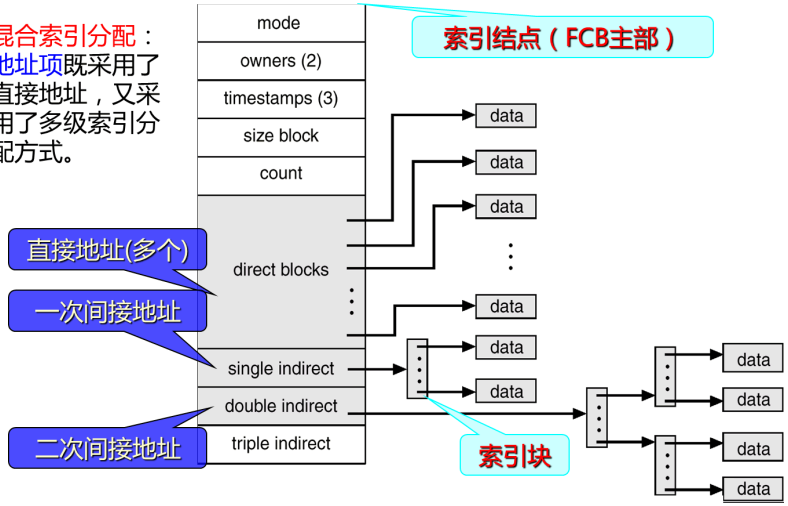
- 在索引分配方式中，每个文件都至少有一个索引块。
- 索引块在外存空间，对于小文件来说，检索效率不高，索引块也消耗了存储空间。
- 由于文件系统 80%以上文件是小文件，为了解决能高速存取小文件和管理大文件的矛盾，UNIX 把直接寻址、一级索引、二级索引和三级索引结合起来，形成了混合索引分配方式。

增量式索引组织方式（混合索引分配）

- 在采用混合索引分配方式的文件系统中，索引结点中设置了 13 个地址项，前 10 个是直接地址项，后 3 个是间接地址项。
- 直接地址：在索引结点中设置 10 个直接地址项，用来存放直接地址。假如每个盘块的大小为 4KB，当文件不大于 40KB 时，便可直接从索引结点中读出该文件的全部盘块号。
- 一次间接地址：利用索引结点中的一次间接地址项来提供一次间址。一次间址块中可存放 1K 个盘块号，允许文件长达 4MB。
- 多次间接地址：利用二次间接地址项提供二次间接地址。在采用二次间址方式时，文件最大长度可达 4GB。同理在采用三次间地方式时，所允许的文件最大长度可达 4TB。

增量式索引组织方式（混合索引分配）

混合索引分配：地址项既采用了直接地址，又采用了多级索引分配方式。



练习题

例：请分别解释在连续分配方式、隐式链接分配方式、显式链接分配方式和索引分配方式中如何将文件的字节偏移量 3500 转换为物理块号和块内位移量（设盘块大小为 1KB，盘块号需占 4 个字节）。

练习题

例：请分别解释在连续分配方式、隐式链接分配方式、显式链接分配方式和索引分配方式中如何将文件的字节偏移量 3500 转换为物理块号和块内位移量（设盘块大小为 1KB，盘块号需占 4 个字节）。

首先，将字节偏移量 3500 转换成逻辑块号和块内位移量：3500 / 1024 得到商为 3，余数为 428，即逻辑块号为 3，块内位移量为 428。

1. 在连续分配方式中，可从相应文件的 FCB 中得到分配给该文件的起始物理盘块号，例如 a0，故字节偏移量 3500 相应的物理盘块号为 a0+3，块内位移量为 428。

练习题

2 在隐式链接方式中，由于每个盘块中需留出 4 个字节（如最后的 4 个字节）来存放分配给文件的下一个盘块的块号，因此字节偏移量 3500 的逻辑块号为 3500 / 1020 的商 3，而块内位移量为余数 440。

从相应文件的 FCB 中可获得分配给该文件的首个（即第 0 个）盘块的块号，如 b0；然后可通过读第 b0 块获得分配给文件的第 1 个盘块的块号 b1；再从 b1 块中得到第 2 块的块号 b2；从 b2 块中得到第 3 块的块号 b3。如此，便可得到字节偏移量 3500 对应的物理块号 b3，而块内位移量则为 440。

练习题

3 在显式链接方式中，可从文件的 FCB 中得到分配给文件的首个盘块的块号，如 c0；然后可在 FAT 的第 c0 项中得到分配给文件的第 1 个盘块的块号 c1；再在 FAT 的第 c1 项中得到文件的第 2 个盘块的块号 c2；在 FAT 的第 c2 项中得到文件的第 3 个盘块的块号 c3。如此，便可获得字节偏移量 3500 对应的物理块号 c3，而块内位移量则为 428。

4 在索引分配方式中，可从文件的 FCB 中得到索引（块）表的地址。从索引表的第 3 项（距离索引表首字节 12 字节的位置）可获得字节偏移量 3500 对应的物理块号 d，而块内位移量为 428。

练习题

例：存放在某个磁盘上的文件系统，采用混合索引分配方式，其 FCB 中共有 13 个地址项（索引项），第 1-10 个地址项为直接地址，第 11 个地址项为一次间接地址，第 12 个地址项为二次间接地址，第 13 个地址项为三次间接地址。如果每个盘块的大小为 512 字节，盘块号需要用 3 个字节来描述，而每个盘块最多存放 170 个盘块地址（512/3≈170）。问：

1. 该文件系统允许文件的最大长度是多少？
2. 将文件的字节偏移量 5000、15000、150000 转换为物理块号和块内偏移量。
3. 假设某个文件的FCB 已在内存，但其他信息均在外存，为了访问该文件中某个位置的内容，最少需要几次访问磁盘，最多需要几次访问磁盘？

练习题

1 该文件系统中一个文件的最大长度可达：

10 + 170 + 170×170 + 170×170×170 = 4942080 块，共 4942080×512 字节 = 2471040 KB ≈ 2.36GB

或用如下解法：

直接索引项可管理上限：10×0.5KB=5KB；

一次间接索引项可管理上限：170×0.5KB=85KB；

二次间接索引项可管理上限：170×170×0.5KB=14450KB；

三次间接索引项可管理上限：170×170×170×0.5KB=2456500KB；

可管理文件上限：5KB+85KB+14450KB+2456500KB。

练习题

2 5000/512 得到商为 9，余数为 392，即字节偏移量 5000 对应的逻辑块号为 9（第 10 块），块内偏移量为 392。由于 9<10，故可直接从该文件的 FCB 的第 10 个地址项处得到物理盘块号，块内偏移量为 392。

15000/512 得到商为 29，余数为 152，即字节偏移量 15000 对应的逻辑块号为 29，块内偏移量为 152。由于 10<29<10+170，而 29-10=19，故可从 FCB 的第 11 个地址项，即一次间址项中得到一次间址块的地址；并从一次间址块的第 19 项中获得对应的物理盘块号，块内偏移量为 152。

练习题

2 150000/512 得到商为 292，余数为 496，即字节偏移量 150000 对应的逻辑块号为 292，块内偏移量为 496。由于 10+170<292<10+170+170×170，而 292-(10+170)=112，112/170 得到商为 0，余数为 112，故可从 FCB 的第 12 个地址项，即二次间址项中得到二次间址块的地址，并从二次间址块的第 1 项中获得一个一次间址块的地址，再从这个一次间址块的第 112 项中获得对应的物理盘块号，块内偏移量为 496。

练习题

3 在采用混合索引分配（三级索引）的系统中，假设某个文件的FCB 已在内存，但其他信息均在外存，为了访问该文件中某个位置的内容，最少需要几次访问磁盘，最多需要几次访问磁盘？

练习题

3 在采用混合索引分配（三级索引）的系统中，假设某个文件的FCB 已在内存，但其他信息均在外存，为了访问该文件中某个位置的内容，最少需要几次访问磁盘，最多需要几次访问磁盘？

由于文件的 FCB 已在内存，为了访问文件中某个位置的内容，最少需要 1 次访问磁盘（即可通过直接地址直接读文件盘块），最多需要 4 次访问磁盘（第一次是读三次间址块，第二次是读二次间址块，第三次是读一次间址块，第四次是读文件盘块）。

文件存储空间的管理

- 文件存储空间的管理可采用连续分配和离散分配方式。
- 内存分配基本不采用连续分配方式。外存管理中，小文件经常采用连续分配方式，大文件采用离散分配方式。

存储空间的管理方法

- 空闲表法和空闲链法
- 位示图法
- 成组链接法

空闲表法

- 空闲表也叫空闲文件目录，是将文件存储器上一个个连续的未分配区域（称作空闲文件）按第一个空闲块号，连续空闲的块数，具体的位置信息（物理块号）等信息记在空闲表中。
- 这种方法适合于建立连续文件，适合少量的空闲区。
- 如果存取空间中有着大量的小的空白区，则空闲表变得很大，因而效率大为降低。

序号	第一空白块号	空白块个数	物理块号
1	2	4	(2 , 3 , 4 , 5)
2	9	3	(9 , 10 , 11)
3	15	5	(15 , 16 , 17 , 18 , 19)
4	—	—	—

空闲表法

- 当请求分配存储空间时，系统依次扫描空闲文件目录的表目，直到找到一个合适的空闲文件为止（首次适应算法），分配给用户进程，修改空闲表。
- 当用户撤消一个文件时，系统回收该文件所占用的空间。扫描空闲表目录，寻找一个空表目，并将释放空间的第一个物理块号及它所占的物理块数填到这个表目中。
- 考虑回收区是否与空闲表中插入点的前区和后区相邻接，对相邻接者应予以合并。

空闲链法

- 空闲盘块链：把所有“空闲块”（即未分配使用的物理块，也称“自由块”）链接起来。当创建一个文件需要存储块时，就从链头上依次取下若干块来，而撤销文件时则将回收空间又依次链接到链尾上。
- 空闲盘区链：把磁盘上的所有空闲盘区（每个盘区可包含若干个盘块）拉成一条链。在每个盘区上除含有用于指示下一个空闲盘区的指针外，还有指明本盘区大小（盘块数）的信息。
- 优点：分配和回收的过程非常简单。
- 缺点：为一个文件分配盘块时，可能要多次重复操作。

位示图法

- 位示图（二维）：利用二进制的一位（bit）来表示磁盘中一个盘块的使用情况。
- 每个字节的每一位都对应了一个物理块的状态。当该位取 1 时，表示对应的物理块已分配；取 0 时表示该物理块未分配。
- 空闲空间表现为位图或位向量。
- 特点：占空间少，可放入内存，易于访问。

位示图法

	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	1	1	0	0	0
2	1	0	1	1	1	0	1	1	1	1
3	0	0	1	0	0	1	0	0	0	0
4	1	1	0	0	0	1	1	1	0	1
5	0	1	1	1	1	0	0	0	1	0
6	1	0	1	1	1	0	1	0	1	1
7	1	1	1	0	0	1	0	0	0	1

位示图法

盘块的分配

1. 顺序扫描，找一个或一组值为 0 的块。
2. 把找到的二进制位转换成与之相应的盘块号。假定找到的“0”的二进制位在第 i 行、第 j 列，每行位数为 n，则相应的盘块号：b=n(i-1)+j

   i,j,b 都从 1 开始
3. 修改位示图：map[i,j]=1。

盘块回收

1. 由盘块号 b 得行列号 (i,j)

   i = (b-1) div n +1

   j = (b-1) mod n +1　　div 取整　mod 取余
2. 修改位图：map[i,j]=0

练习题

例：设某系统的磁盘有 500 块，块号为：0，1，2，3，...，499。

（1）若用位示图法管理这 500 块的盘空间，当字长为 32 位时，此位示图占了几个字？

（2）第 i 字的第 j 位对应的块号是多少？（其中：i=0，1，2，...，j=0，1，2，3，...）

练习题

例：设某系统的磁盘有 500 块，块号为：0，1，2，3，...，499。
（1）若用位示图法管理这 500 块的盘空间，当字长为 32 位时，此位示图占了几个字？
（2）第 i 字的第 j 位对应的块号是多少？（其中：i=0，1，2，...，j=0，1，2，3，...）

1. 位示图法就是在内存用一些字建立一张位示图，用其中的每一位表示一个盘块的使用情况，通常用“1”表示占用，“0”表示空闲。因此，本问题中位示图所占的字数为：500 / 32 = 15.625 ≈ 16。
2. 第 i 字的第 j 位对应的块号 N = 32 * i + j。

练习题

例：有一计算机系统利用下图所示的位示图（行号、列号都从 0 开始编号）来管理空闲盘块。如果盘块从 1 开始编号，每个盘块的大小为 1KB。

（1）现要为文件分配两个盘块，试具体说明分配过程。

（2）若要释放磁盘的第 300 块，应如何处理？

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5																
6																

练习题

1 为某文件分配两个盘块的过程如下：

①顺序检索位示图，从中找到第一个值为 0 的二进制位，得到其行号 i1=2，列号 j1=2。第二个值为 0 的二进制位，得到其行号 i2=3，列号 j2=6。

②计算出找到的两个空闲块的盘块号分别为：

b1 = i1×16 + j1 + 1 = 2×16 + 2 + 1 = 35

b2 = i2×16 + j2 + 1 = 3×16 + 6 + 1 = 55

③修改位示图，令 map[2,2]=map[3,6]=1，并把对应的块 35 和块 55 分配出去。

练习题

2 释放磁盘的第 300 块时，应进行如下处理：

①计算出磁盘第 300 块所对应的二进制位的行号 i 和列号 j：

i = (300 - 1) div 16 = 18

j = (300 - 1) mod 16 = 11

②修改位示图，令 map[18,11]=0，表示对应块为空闲块。

位示图法

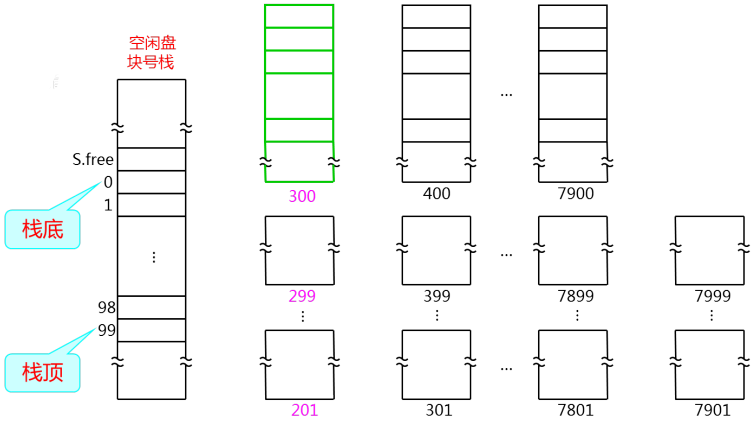
- i,j,b 都从 1 开始
  - 盘块号：b=n(i-1)+j
  - 行列号：i = (b-1) div n +1 ；j = (b-1) mod n +1
- i,j,b 都从 0 开始
  - 盘块号：b=n×i+ j
  - 行列号：i = b div n ；j = b mod n
- i,j 都从 0 开始，b 从 1 开始
  - 盘块号：b=n×i+ j+1
  - 行列号：i = (b-1) div n ；j = (b-1) mod n
- i,j 都从 1 开始，b 从 0 开始
  - 盘块号：b=n(i-1)+j-1
  - 行列号：i = b div n +1 ；j = b mod n +1

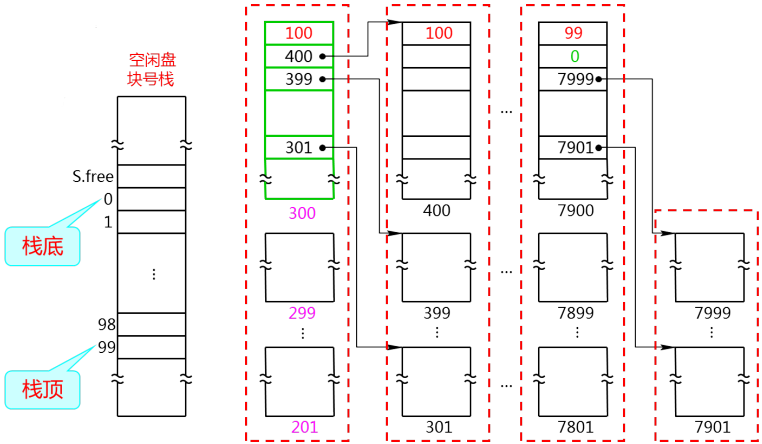
成组链接法的引入

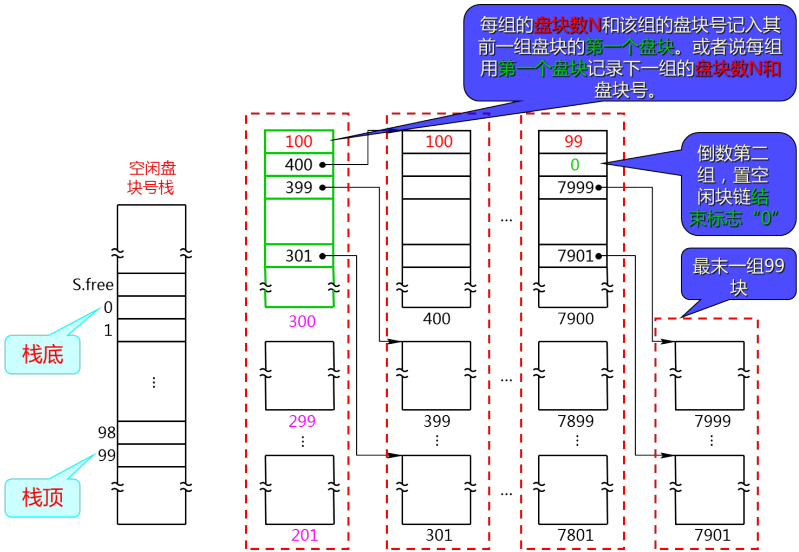
- 在文件存储空间的管理中，空闲表法属于连续分配方式，系统为外存上的所有空闲区建立一张空闲表，每个空闲区对应一个空闲表项。分配盘块时通常采用类似于内存分配的首次适应算法、循环首次适应算法。空闲表法的分配和回收效率较高。
- 空闲链法属于离散分配方式。空闲链法分配空闲盘块或盘区更加简单，但效率不高，可能要多次重复操作。
- 成组链接法是结合了空闲表和空闲链优点的一种空闲盘块管理方法，并克服了空闲表过大或空闲链过长的缺点。

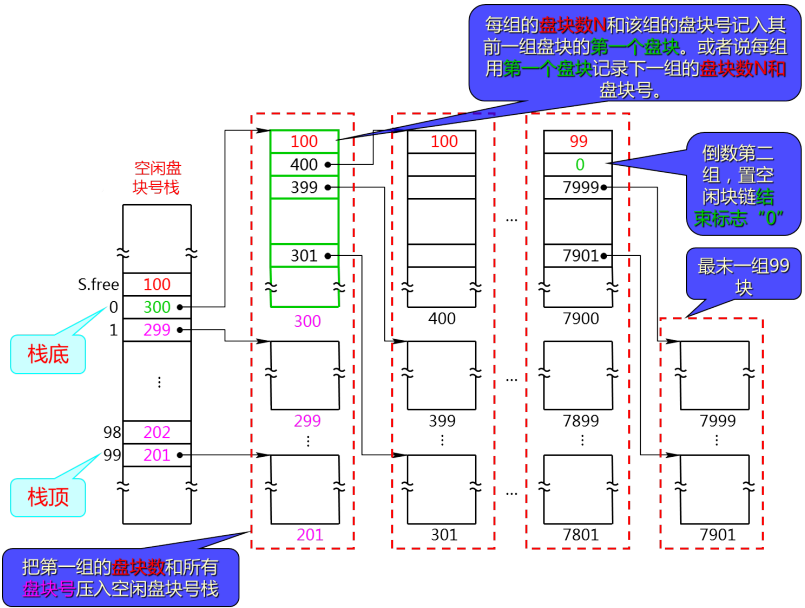
空闲盘块的组织

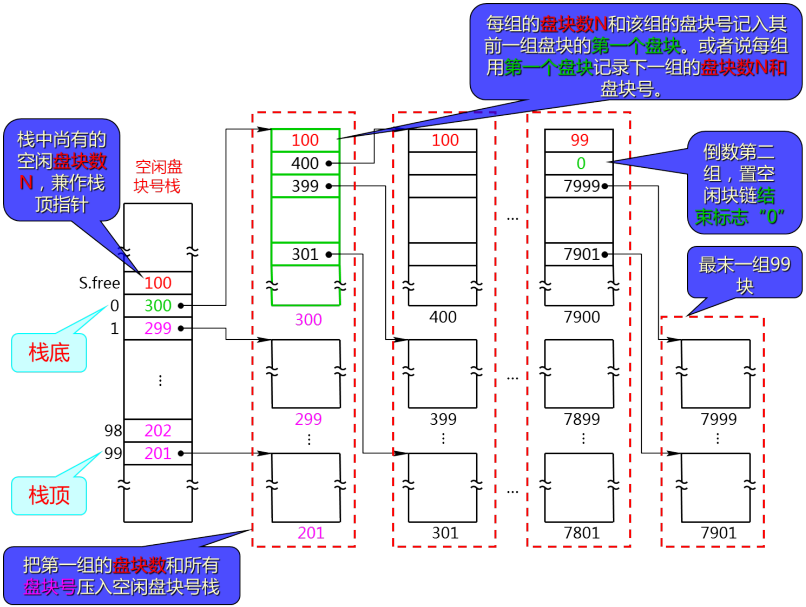
- 空闲盘块号栈：存放一组空闲盘块的盘块号（最多 100 个），以及栈中尚有的空闲盘块号数 N（兼作栈顶指针）。s.free(0) 是栈底，栈满时的栈顶为 S.free(99)。
- 所有空闲盘块被分成若干组（每组 100 个）。
- 把每组的盘块数 N 和该组的盘块号记入其前一组盘块的第一个盘块。
- 把第一组的盘块数和所有盘块号记入空闲盘块号栈。
- 最末一组置空闲块链的结束标志。


空闲盘
块号栈
S.free
0
1
98
99
栈底
栈顶
300
400
7900
299
399
7899
7999
201
301
7801
7901


空闲盘
块号栈
S.free
0
1
98
99
栈底
栈顶
100
400
399
301
300
299
201
100
400
399
301
99
0
7999
7901
7900
7899
7801
7999
7901


每组的盘块数N和该组的盘块号记入其前一组盘块的第一个盘块。或者说每组用第一个盘块记录下一组的盘块数N和盘块号。
空闲盘块号栈
S.free
0
1
98
99
栈底
栈顶
100
400
399
301
300
299
201
100
400
301
99
0
7999
7901
7900
7899
7801
7999
7901
倒数第二组，置空闲块链结束标志“0”
最末一组99块

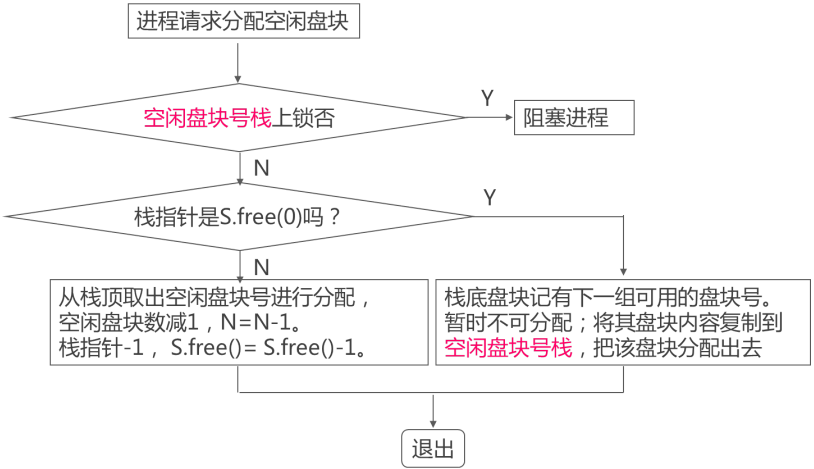

每组的盘块数N和该组的盘块号记入其前一组盘块的第一个盘块。或者说每组用第一个盘块记录下一组的盘块数N和盘块号。
空闲盘块号栈
S.free
100
0
300
1
299
98
202
99
201
栈底
栈顶
100
400
399
301
300
299
201
100
400
399
301
99
0
7999
7901
7900
7899
7801
7999
7901
倒数第二组，置空闲块链结束标志“0”
最末一组99块
把第一组的盘块数和所有盘块号压入空闲盘块号栈


每组的盘块数N和该组的盘块号记入其前一组盘块的第一个盘块。或者说每组用第一个盘块记录下一组的盘块数N和盘块号。
栈中尚有的空闲盘块数N，兼作栈顶指针
空闲盘块号栈
倒数第二组，置空闲块链结束标志“0”
最末一组99块
S.free
0
1
98
99
100
300
299
202
201
栈底
栈顶
把第一组的盘块数和所有盘块号压入空闲盘块号栈
100
400
399
300
299
201
100
400
399
301
99
0
7999
7901
7900
7899
7801
7999
7901

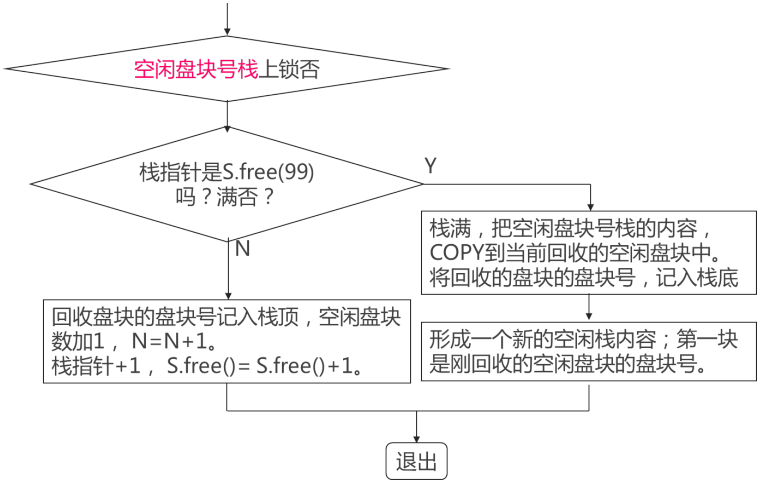
空闲盘块的分配与回收

- 分配：从栈顶取出一空闲盘块号，将与之对应的盘块分配给用户，然后将栈顶指针下移一格。到s.free(0)时，所分配的磁盘块号是栈中最后一个可用盘块号，由于该块内容为下一组的所有盘块号，因此要先将该块的内容读入栈中，然后才能将该块分配出去。
- 回收：将回收盘块的盘块号压入空闲盘块号栈的顶部，并执行空闲盘块数 N加 1 操作。到栈满时，将现有栈中的 100 个盘块弹出栈，组成新的一组空闲盘块组。它们的盘块号被记入新回收的盘块中，再将其盘块号作为新栈底（记录了前一组的 100 个盘块号）。

空闲盘块的分配


进程请求分配空闲盘块
空闲盘块号栈上锁否
Y
阻塞进程
N
栈指针是S.free(0)吗？
Y
N
从栈顶取出空闲盘块号进行分配，
空闲盘块数减1，N=N-1。
栈指针-1， S.free()= S.free()-1。
栈底盘块记有下一组可用的盘块号。
暂时不可分配；将其盘块内容复制到
空闲盘块号栈，把该盘块分配出去
退出

空闲盘块的回收


空闲盘块号栈上锁否
栈指针是S.free(99)
吗？满否？
Y
N
栈满，把空闲盘块号栈的内容，
COPY到当前回收的空闲盘块中。
将回收的盘块的盘块号，记入栈底
回收盘块的盘块号记入栈顶，空闲盘块
数加1， N=N+1。
栈指针+1， S.free()= S.free()+1。
形成一个新的空闲栈内容；第一块
是刚回收的空闲盘块的盘块号。
退出

成组链接法

- 成组链接法的优点：
  - 空白块号登记不占用额外空间，只临时借用每组的第一个空白块（读块后仍可以分配给用户用）。
  - 当前可分配的物理块号存放在空闲盘块栈，因此绝大部分的分配和回收工作是在主存中进行，可以节省时间。

提高磁盘 I/O 速度的途径

提高文件的访问速度

1. 改进文件的目录结构以及检索目录的方法以减少对目录的查找时间。
2. 选用好的文件存储结构
3. 提高磁盘 I/O 速度
   - 设置磁盘高速缓存（Disk Cache）
   - 其它方法（提前读、延迟写、虚拟盘）
   - 采用高度可靠、快速的磁盘系统：独立磁盘冗余阵列（RAID）

磁盘高速缓存

- 磁盘高速缓存是指内存中的一部分存储空间，用来暂存从磁盘读出的一系列盘块中的信息，所以它是一组在逻辑上属于磁盘，而物理上是驻留在内存中的盘块。
- 磁盘高速缓存的形式
  - 内存中单独的存储空间。（大小固定）
  - 未利用的内存存储空间：缓冲池。（大小不固定）
- 磁盘高速缓存与请求分页系统共享缓冲池。

数据交付方式

- 数据交付是指将磁盘高速缓存中的数据传送给请求者进程。
- 步骤：请求访问磁盘时先查缓存、后查磁盘并更新缓存。
- 系统采取两种方式把数据交付给请求进程：
  - 数据交付：系统直接将磁盘高速缓存中的数据传送到请求者进程的内存工作区。
  - 指针交付：只将指向磁盘高速缓存中该数据的指针，交付给请求者进程。
- 后一种方式由于所传送的数据量少，节省了数据从磁盘高速缓存到进程的内存工作区的时间。

置换算法

- 在将磁盘中的盘块读入到磁盘高速缓存中时，若因磁盘高速缓存已满，则采用常用的算法进行置换：
  - 最近最久未使用算法 LRU
  - 最近未使用算法 NRU
  - 最少使用算法 LFU
- 请求调页与磁盘 I/O 中的工作情况不同，在置换算法中所应考虑的问题也有所差异。磁盘高速缓存置换时除上述算法外还应考虑的问题：
  - 访问频率：请求调页 > 磁盘 I/O
  - 可预见性：请求调页 < 磁盘 I/O
  - 数据的一致性：一旦系统发生故障，高速缓存中的数据将会丢失，如果未拷回磁盘，会造成数据的不一致。

周期性地写回磁盘

- 数据的一致性解决方法：将系统中所有盘块数据拉成一条 LRU（最近最久未使用算法）链，需要一致性的块放在 LRU 队列的头部，优先回写。可能在不久之后再使用的盘块数据，挂在 LRU 链的尾部。
- 系统中所有盘块数据拉成一条 LRU 链，经常访问的数据被挂在链尾，一直保留磁盘高速缓存中，可能长期不会被写回磁盘。若系统出故障，则存在磁盘高速缓存中的数据将丢失。
- 解决方案：① 周期性地将磁盘高速缓存中的数据写回磁盘。② 数据若有修改则立即写回磁盘 (写穿透，write through cache) 。缺点是需要频繁启动磁盘。

提高磁盘 I/O 速度的其它方法

- 提前读（Read-Ahead）
  - 由于用户对文件的访问常用顺序方式，所以可采用预先读方式，即在读当前块的同时，连同将下一块提前读入缓冲。当访问下一块数据时，其已在缓冲中，而不需去启动磁盘 I/O。
- 延迟写（Write-Behind）
  - 在缓存中的数据，本应立即写回磁盘，考虑不久可能再用，故不立即写回磁盘，挂在空闲缓冲区队列末尾。
  - 当该缓冲区仍在队列中时，任何访问该数据的进程，都可直接读出其中的数据而不必去访问磁盘。

提高磁盘 I/O 速度的其它方法

- 优化物理块的分布
  - 使磁头移动的距离最小。
  - 多个连续的块组成一簇，以簇为单位进行分配。
- 虚拟盘
  - 虚拟盘是利用内存去仿真磁盘，又称为 RAM 盘。
  - 虚拟盘对用户是透明的，仅是感觉略微快些而已。
  - 虚拟盘是易失性存储器（内存），关机或重新启动计算机时，虚拟盘上的信息将丢失，所以在关机或重新启动计算机前，一定要及时把在虚拟盘上的重要的数据存放到真正的硬盘中。
  - 虚拟盘与磁盘高速缓存的区别：RAM 盘中的内容由用户控制，而缓存中的内容则由 OS 控制。

廉价磁盘冗余阵列（RAID）

- RAID (Redundant Array of Inexpensive Disks)
- RAID (Redundant Array of Independent Disks)
- RAID是利用一台磁盘阵列控制器来统一管理和控制一组磁盘驱动器，组成一个高度可靠的、快速的大容量磁盘系统。
- 并行交叉存取（条带化存取）
- 冗余存取
- 校验存取
- 优点：可靠性高、磁盘 I/O 速度高、性价比高

廉价磁盘冗余阵列（RAID）

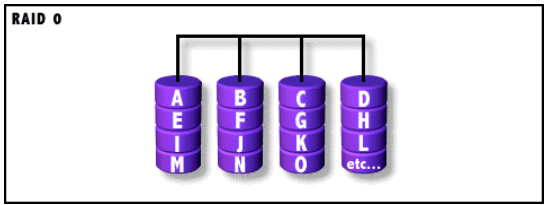
- RAID 的两种工作方式
  - 同步：阵列中所有驱动器主轴同步旋转，数据按位/字节组织，各驱动器同时存取数据中的 1 位/1 个字节。
  - 异步：阵列中各驱动器主轴不同步，数据按块组织，各驱动器独立存取同一文件中各块数据。
- 同步磁盘阵列控制复杂，成本高。
- 异步磁盘阵列需要较大容量的缓存。

并行交叉存取

- 并行交叉存取技术：为提高磁盘的访问速度而在大中型机中采用的磁盘技术。
- 系统将每一盘块的数据分成若干个子盘块数据，再把每一个子盘块的数据分别存储到各个不同磁盘中的相同位置。
- 当要将一个盘块的数据传送到内存时，采取并行传输方式，将该盘块中的各个子盘块数据同时向内存传输。
- 在系统中，RAID 磁盘组（最少两块）被用户看成是一个硬盘，用户可以对它进行分区，格式化等，对磁盘阵列的操作与单个硬盘一模一样。

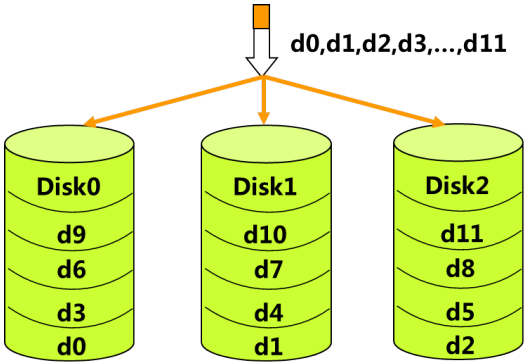
RAID 0

- Striped Disk Array without Fault Tolerance（没有容错设计的条带磁盘阵列）
- 将数据分条（stripping），并存于不同磁盘上。因多个磁盘可同时工作，数据传输效率高。
- 仅提供了并行交叉存取，没有数据冗余；无容错功能，不增强可靠性。
- RAID 0 不算是真正的 RAID，它没有数据冗余能力。



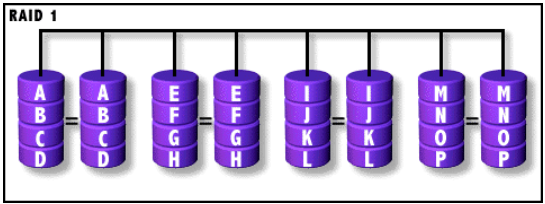
RAID 0

- 例如，应用程序读取 d6、d7 和 d8，RAID 控制器将该请求分解为三个命令，各对应一个磁盘驱动器，使其并行工作，这种并行没有软件的参与。



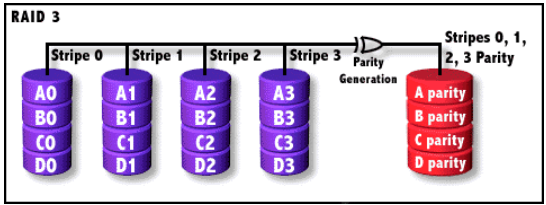
RAID 1

- Mirroring and Duplexing（相互镜像）
- 硬盘的内容两两相同，即存在镜像（mirroring）。完全相同的数据重复存于多个盘上。注意：数据读取时可选任一独立磁盘，但修改时必须对所有镜像同时进行，以保证数据一致性。
- 具有高可靠性，分布存放；镜像冗余，空间利用率 50%，代价很高，需要 2n 个独立磁盘；不校验。



RAID 2

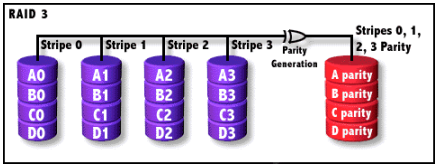
- Hamming Code ECC（汉明码错误检测与修正）
- 所有磁盘是同步的，即任何时刻每个磁头都位于各自磁盘的相同位置。
- 对每个磁盘相应位都计算一个错误校正码，保存于多个奇偶校验磁盘的相应位中。错误校正采用汉明码，可纠正一位错误，检测两位错误。
- 需要多个磁盘存放校验信息，技术复杂，未广泛应用。



RAID 3

- Parallel transfer with parity（并行传输及校验）
- RAID 3 是在 RAID 2 基础用相对简单的异或逻辑运算（XOR, eXclusive OR ）校验代替了汉明码校验。
- RAID 3 采用并行传输及校验。不论有多少数据盘，均使用一个校验盘，数据分成条带存入数据阵列中。
- 在数据存入时，数据阵列中处于同一等级的条带的 XOR 校验编码被即时写在校验盘相应的位置。读取时，则在调出条带的同时检查校验盘中相应的 XOR 编码。
- 任何一个单独的磁盘驱动器损坏，奇偶校验盘及其他数据盘可以重新恢复数据。

RAID 3



A 值	B 值	XOR 结果
0	0	0
1	0	1
0	1	1
1	1	0

例：一个5个驱动器的阵列，包括4个数据盘$X0 \sim X3$和一个校验盘$X4$。

第i位的奇偶校验可计算如下：

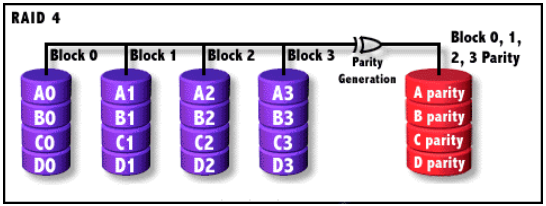
$$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$$

假如$X1$失败，用其余三块盘与$X4$做异或：

$$X1(i) = X4(i) \oplus X3(i) \oplus X2(i) \oplus X0(i)$$

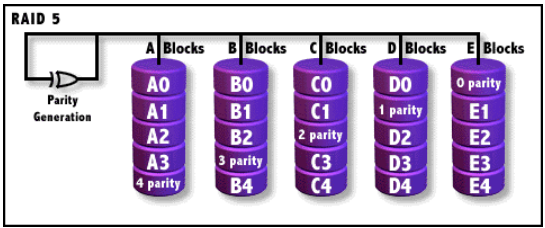
RAID 4

- Independent Data disks with shared Parity disk（独立的数据硬盘与共享的校验硬盘）
- RAID 4 和 RAID 3 很相似，不同的是 RAID 4 对数据的访问是按数据块进行的。比如一个文件就是一个数据块，按块存储可以保证数据的完整性。
- RAID 3 和 RAID 4 容易造成校验磁盘的瓶颈。



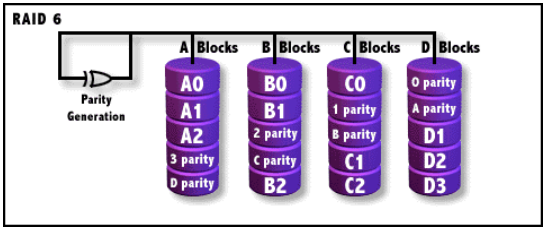
RAID 5

- Independent Data disks with distributed parity blocks（独立的数据磁盘与分布式校验块）
- RAID 5 不用校验磁盘而将校验数据以循环的方式放在每一个磁盘中。避免了 RAID 4 的瓶颈，提高了可靠性。
- RAID 5 的控制比较复杂，但磁盘利用率高，若任意一个磁盘发生故障，可根据其它 N-1 个磁盘进行恢复，是最常见的磁盘阵列组织形式。



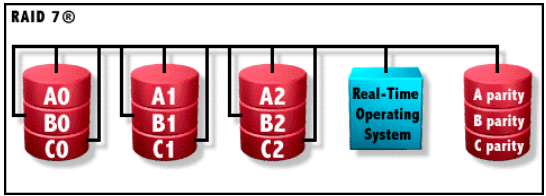
RAID 6

- Independent Data disks with two independent distributed parity schemes（独立的数据硬盘与两个独立分布式校验方案）
- RAID 6 除了每个硬盘上都有同级数据 XOR 校验区外，还有一个针对每个数据块的 XOR 校验区 (双重校验)。数据块的校验数据交错存储。
- RAID 6 能在两块硬盘同时发生故障的情况下修复数据。



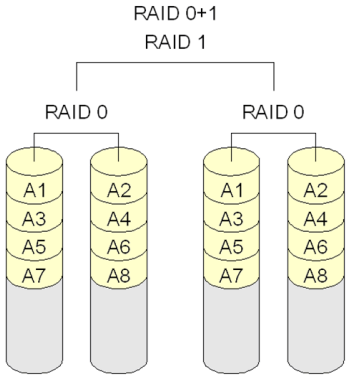
RAID 7

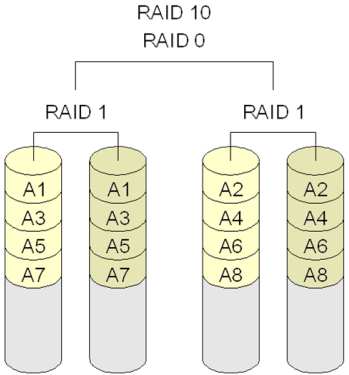
- Optimized Asynchrony for High I/O Rates as well as High Data Transfer Rates（最优化的异步高 I/O 速率和高数据传输率）
- RAID 7 自身带有智能化实时操作系统和用于存储管理的软件工具，可完全独立于主机运行，不占用主机 CPU 资源。
- RAID 7 是至今为止理论上性能最高的 RAID 模式。



RAID 10

- 此级别是 RAID 0 与 RAID 1 的结合（stripping+mirroring）





各级 RAID 的比较

RAID级别	描述	速度	容错性能	其它
RAID 0	分条	并行I/O	无	
RAID 1	磁盘镜像	没有提高	允许单个磁盘错误	空间利用率50%
RAID 2	分条+汉明码纠错	并行I/O	允许单个磁盘错误	
RAID 3	分条+专用校验盘	并行I/O	允许单个磁盘错误	按条带校验
RAID 4	分条+专用校验盘	并行I/O	允许单个磁盘错误	按块校验
RAID 5	分条+分布式校验	并行I/O	允许单个磁盘错误	最常见的RAID
RAID 6	分条+分布式条带校验+分布块校验	并行I/O	允许两个磁盘错误	
RAID 7	自带智能化实时OS	并行I/O	很好	完全独立于主机

提高磁盘可靠性的技术

- 影响文件安全性的主要因素
  - 人为因素
  - 系统因素
  - 自然因素
- 通过存取控制机制来防止由人为因素所造成的文件不安全性。（访问矩阵）
- 通过磁盘容错技术，来防止由磁盘部分的故障所造成的文件不安全性。
- 通过后备系统来防止由自然因素所造成的不安全性。

磁盘容错技术

- 容错技术是通过在系统中设置冗余部件的办法，来提高系统可靠性的一种技术。
- 磁盘容错技术：通过增加冗余的磁盘驱动器、磁盘控制器等方法来提高磁盘系统可靠性的一种技术。又称为系统容错技术 SFT（System Fault Tolerance）。

磁盘容错技术

1. 第一级磁盘容错技术 SFT-I
2. 第二级磁盘容错技术 SFT-II
3. 基于集群技术的容错功能 SFT-III

第一级磁盘容错技术 SFT-I

第一级磁盘容错技术 SFT-I主要用于防止因磁盘表面发生缺陷所引起的数据丢失。采用的措施有：

1. 双份目录和双份文件分配表
   - 分别建立双份目录表和文件分配表（FAT）。
   - 在系统每次启动时都要进行两份目录和分配表的检查。

第一级磁盘容错技术 SFT-I

2 热修复重定向和写后读校验

- 热修复重定向（Hot-Fix Redirection）
- 系统将一定的磁盘容量（如 2%-3%）作为热修复重定向区。
- 例如：系统要向第 3 柱 2 头 1 扇区写数据，但发现该扇区是坏的时，便将数据写到热修复区（如 200 柱 16 头 1 扇区）。以后要读 3 柱 2 头 1 扇区的数据时，便从 200 柱 16 头 1 扇区中读。

第一级磁盘容错技术 SFT-I

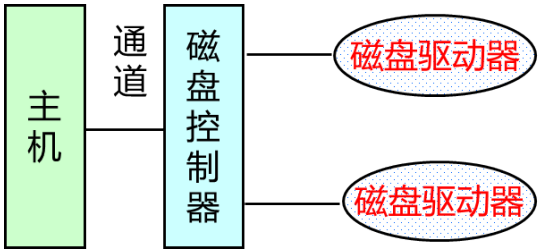
2 热修复重定向和写后读校验

- 写后读校验(Read after Write Verification)
- 为了保证所有写入到磁盘的数据都能写入完好的盘块中，在每次写数据时，又立即从磁盘上读出该块数据，并同写前的数据进行对比（校验）。
- 若两者不一致，重写，继续比较。
- 若仍不一致，则认为盘块有缺陷，将该数据写入到热修复区。
- 对该坏盘块进行登记。

第二级磁盘容错技术 SFT-II

第二级磁盘容错技术 SFT-II用于防止由磁盘驱动器和控制器的故障所导致的系统不能正常工作。具体又分为磁盘镜像和磁盘双工。

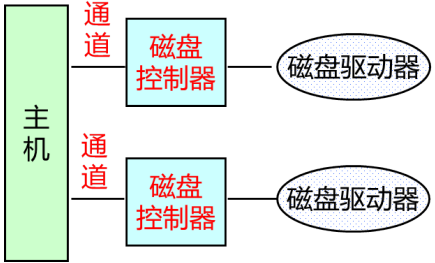
1. 磁盘镜像 (Disk Mirroring)
   - 同一磁盘控制器下，增设一个完全相同的磁盘驱动器。
   - 磁盘 I/O 速度下降，利用率降至 50%。



第二级磁盘容错技术 SFT-II

2 磁盘双工 (Disk Duplexing)

- 将两台磁盘驱动器分别接到两个磁盘控制器上，同时使这两台磁盘机镜像成对。
- 两个硬盘子系统互为镜像。
- 分离搜索技术 (Split Seek)：从响应快的通道上取数据。



集群 (Cluster) 技术

- 集群 (Cluster) 技术指由一组互连的自主计算机组成统一的计算机系统，给人们的感觉是，它们是一台机器。
- 一个集群包含多台（至少二台）拥有共享数据存储空间的服务器。集群内任一系统上运行的服务可被所有的网络客户所使用。
- 可以采用多台 SMP（对称多处理技术）服务器来实现集群系统服务器。
- 利用集群系统不仅可提高系统的并行处理能力，还可用于提高系统的容错功能。当一台节点服务器发生故障时，这台服务器上所运行的应用程序将在另一节点服务器上被自动接管。

基于集群技术的容错功能 SFT-III

- 基于集群技术的容错功能 SFT-III利用集群系统提高系统的容错性能（可用性）。
- 可防止以下机器失效带来的损失：随机存储器 (RAM) 的失效、磁盘的失效、局域网适配器的失效。

基于集群技术的磁盘容错技术

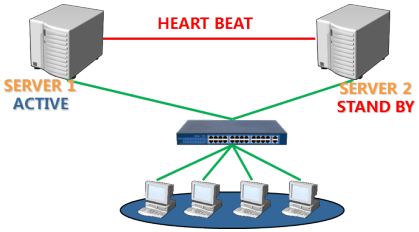
1. 双机热备份模式
2. 双机互为备份模式
3. 公用磁盘模式

双机热备份模式 (Hot Standby)

- Active/Standby 方式
- 热备份：在线的备份称为热备份; 脱机数据备份称为冷备份。
- 在这种模式的系统中，备有两台服务器，两者的处理能力通常是完全相同的，一台作为主服务器 (Active)，另一台作为备份服务器 (Standby)。
- 平时主服务器运行，备份服务器则时刻监视着主服务器的运行，一旦主服务器出现故障，备份服务器便立即接替主服务器的工作而成为系统中的主服务器，修复后的服务器再作为备份服务器。

双机热备份模式 (Hot Standby)

- 每个服务器通过高性能镜像服务器链路 MSL (Mirrored Server Links) 连接。MSL 如果采用光纤链路，距离可以达到 20-100 公里。
- 双机热备份技术系统采用“心跳”(Heart Beat) 方法保证主系统与备用系统的联系。所谓“心跳”，是指主从系统之间相互按照一定的时间间隔发送通讯信号，表明各自系统当前的运行状态。



双机热备份模式 (Hot Standby)

- Active 服务器处于工作状态，而 Standby 服务器处于监控准备状态。
- 服务器数据同时往两台或多台服务器写入，保证数据的即时同步。
- 优点：易于实现；主、从服务器完全独立；可实现远程热备份。
- 缺点：从服务器长期处于被动等待的状态，系统资源存在一定的浪费。

双机互为备份模式

- App1/App2 方式
- 两台服务器均为在线服务器，它们各自完成自己的任务。例如，一台作为数据库服务器，另一台作为电子邮件服务器。为了实现两者互为备份，两台服务器通过磁盘阵列或纯软件模式，连接成为互为备份的双机系统。
- 在每台服务器内都配置两台硬盘，一个用于装载系统程序和应用程序，另一个用于接收由另一台服务器发来的备份数据，作为该服务器的镜像盘。在正常运行时，镜像盘对本地用户锁死的。
- 两台服务器需要通过专线连接（心跳线）。

双机互为备份模式

- 两台服务器作为一个整体对网络提供服务，且相互监控。集群具有一定的负载平衡功能，可将一个任务的多个进程分摊到两台服务上运行，提高系统整体性能。
- 当一台服务器发生故障时，所运行的进程及服务可以自动地由另一台服务器接管，保证用户的工作不受影响。
- 双机互备模式可以从两台机器扩大到 4 台、8 台、16 台甚至更多（多机互备）。
- 优点：双机皆跑应用，不浪费机器。
- 缺点：双机压力可能互不相同，当全转到一套机器上时，该机的承受能力可能出现问题，如果配置过高又造成浪费。

双机双工模式

- Active/Active 方式
- 两台或多台服务器均处于工作状态，同时运行相同的应用程序，保证了整体的性能，也实现了负载均衡和互为备份。

双机双工模式

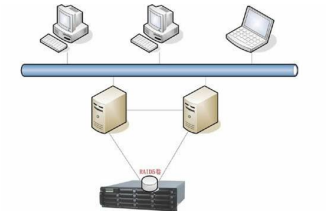
- Active/Active 方式
- 两台或多台服务器均处于工作状态，同时运行相同的应用程序，保证了整体的性能，也实现了负载均衡和互为备份。

双机容错模式

1. Power on/Power off 方式：双机冷备
2. Active/Standby 方式：双机热备（硬件冗余，手动切换）
3. App1/App2 方式：双机互备（硬件冗余，自动切换）
4. Active/Active 方式：双机双工（性能冗余，自动切换）

公用磁盘模式

- 为了减少信息复制（备份）的开销，可以将多台计算机连接到一台公共的磁盘系统上去。
- 基于共享磁盘阵列模式的双机集群系统，使用磁盘阵列作为两台服务器的共用存储设备，通过在两台服务器上运行高可靠性软件（双机软件或集群软件）对磁盘阵列进行管理，同时对受保护的服务进行监控和管理。

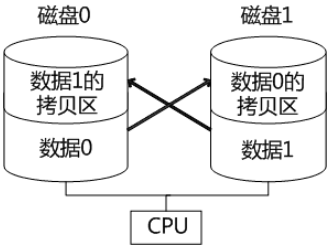


公用磁盘模式

- 优点
  - 实现真正意义上的数据与系统分离，系统整体效率高，存储系统升级扩容方便。
  - 消除了信息的复制时间，因而减少了网络和服务器的开销。
- 缺点
  - 存在单点数据故障的可能。一旦磁盘阵列出现逻辑或物理故障，数据安全就得不到保障。

后备系统

- 硬盘：利用大容量硬盘兼做后备系统。每个硬盘分为两个区：数据区和备份区。
- 光盘：CD、DVD、BD



数据一致性控制

- 问题：
  - 在数据应用中，如果需要把一个数据分别存储到多个文件中，可能将会出现同一数据不一致性。
- 解决方案
  - 软件：配置能保证数据一致性的软件。
  - 硬件：采用冗余技术，使用高度可靠的磁盘系统（稳定存储器）。
- 保证数据一致性的软件手段：
  - 事务
  - 检查点

事务

- 事务是用于访问和修改各种数据项的一个程序单位，可被看成是一系列的读和写操作。
- 事务操作是原子操作，对分布在不同位置的同一数据进行的读和写操作全部完成后，才能再以托付操作 (Commit Operation) 来终止事务。
- 只要有一个读、写或修改操作失败，便须执行夭折操作 (Abort Operation) ，已修改的部分也全部恢复原来的情况 (Rolled Back)。
- 事务操作的原子性借助于存放在稳定存储器中的事务记录来实现。

事务

事务的四大属性 ACID

1. 原子性（Atomicity）
   原子性是指事务是一个不可分割的工作单位。
2. 一致性（Consistency）
   事务完成时必须使数据保持一致性状态。
3. 隔离性（Isolation）
   事务对数据的修改必须与其他并发事务相隔离。
4. 持久性（Durability）
   事务一旦被提交，它对系统的影响是永久性的。

事务记录 (Transaction Record)

- 事务记录用来记录在事务运行时数据项修改的全部信息，又称为运行记录 (Log)。该记录中包括有下列字段:
  - 事务名：用于标识该事务的唯一名字；
  - 数据项名：它是被修改数据项的唯一名字；
  - 旧值：修改前数据项的值；
  - 新值：修改后数据项将具有的值。
- 每条记录描述了事务运行中的重要操作，如修改操作、开始操作、托付事务和夭折事务等。
  - 事务 Ti 开始时，〈Ti 开始〉记录被写入事务记录表中；
  - Ti 执行期间，在任何写 (修改) 操作之前，写一新记录到事务记录表中；
  - Ti 进行托付时，〈Ti 托付〉记录被写入事务记录表中。

恢复算法

- 被事务 Ti 修改的数据以及它们被修改前和修改后的值都能在事务记录表中找到，因此利用事务记录表，系统能处理任何故障而不造成存储器中信息的丢失。
- 恢复算法可利用以下两个过程：
  - undo〈Ti〉撤销：该过程把所有被事务 Ti 修改过的数据，恢复为修改前的值。
  - redo〈Ti〉恢复：该过程能把所有被事务 Ti 修改过的数据，设置为新值。
- 如果系统发生故障，系统应对以前所发生的事务进行清理。

恢复算法

- 如果系统发生故障，通过查找事务记录表，可以把尚未清理的事务分成两类：
  - 一类是其所包含的各类操作都已完成的事务。确定为这一类事务的依据是，在事务记录表中，既包含了〈Ti 开始〉记录，又包含了〈Ti 托付〉记录。此时系统利用 redo〈Ti〉过程，把所有已被修改的数据设置成新值。
  - 另一类是其所包含的各个操作并未全部完成的事务。对于事务 Ti，如果在 Log 表中只有〈Ti 开始〉记录而无〈Ti 托付〉记录，则此 Ti 便属于这类事务。此时，系统便利用 undo〈Ti〉过程，将所有已被修改的数据，恢复为修改前的值。

检查点

- 系统发生故障时，必须去检查整个 Log 表。由于在系统中可能存在着许多并发执行的事务，随着时间的推移，记录的数据也会越来越多。
- 引入检查点的主要目的，是使对事务记录表中事务记录的清理工作经常化。
- 每隔一定时间便将驻留在内存中的当前事务记录表中的所有记录、已修改的数据、〈检查点〉记录输出到稳定存储器中。
- 每当出现一个〈检查点〉记录时，系统便执行恢复操作，利用 redo 和 undo 过程实现恢复功能。

检查点

- 如果一个事务 Ti 在检查点前就做了托付，以后在系统出现故障时，就不必再执行 redo 操作了。
- 新的恢复算法
  - 只需对最后一个检查点之后的事务记录进行处理。
  - 如果在事务记录表中出现了 <Tk 托付 > 记录，则执行 Redo 操作；
  - 如果在事务记录表中未出现 <Tk 托付 > 记录，则执行 Undo 操作。

并发控制

- 由于事务的原子性，各个事务的执行必然是按某种次序依次执行的，只有在一个事务执行完后，才允许另外一个事务执行，即事务对数据项的修改是互斥的。可把这种特性称为顺序性（Serializability）。
- 用于实现事务顺序性的技术称为并发控制（Concurrent Control）。

1. 利用信号量机制实现顺序性
2. 利用互斥锁实现顺序性
3. 利用互斥锁和共享锁实现顺序性

利用互斥锁 (Exclusive Lock) 实现顺序性

- 为每一个对象（共享文件、记录或数据项）设置一个用于实现互斥的锁。
- 事务 Ti 要去访问某对象时，应先获得该对象的互斥锁。若成功，将该对象锁住，执行读或写操作；如果 Ti 需要对一批对象进行访问，Ti 应先获得这一批对象的互斥锁，全部锁住。操作完成后把所有的锁全部释放。
- 如果这一批对象中的某一个已被其它事务锁住，Ti 应对所有对象进行开锁，此次事务运行失败，但不致引起数据的变化。

利用互斥锁和共享锁实现顺序性

- 利用互斥锁来锁住文件后，只允许一个事务读写，无法实现共享。而共享文件虽然只允许一个事务去写，但应该允许多个事务同时去读。
- 为了提高运行效率，又引入了另一种形式的锁：共享锁(Shared Lock)。
- 共享锁与互斥锁的区别
  - 互斥锁仅允许一个事务对相应对象执行读或写操作。
  - 共享锁则允许多个事务对相应对象执行读操作，不允许其中任何一个事务对对象执行写操作。

利用互斥锁和共享锁实现顺序性

- 如果事务 Ti 要对 Q 执行读操作，则只需去获得对象 Q 的共享锁。如果对象 Q 已被互斥锁锁住，则 Ti 必须等待；否则便可获得共享锁而对 Q 执行读操作。
- 如果 Ti 要对 Q 执行写操作，则 Ti 必需获得 Q 的互斥锁。若失败，则 Ti 必须等待；否则可获得互斥锁而对 Q 执行写操作。
- 利用共享锁和互斥锁来实现顺序性的方法，类似于第二章中的读者—写者问题的解法。

重复数据的数据一致性问题

重复数据的数据一致性检查

1. 重复文件的一致性检查
2. 链接数的一致性检查
3. 盘块号的一致性检查

重复文件的一致性检查

- 为保证文件系统的可用性，在有些系统中为关键文件设置了多个重复拷贝，把它们分别存储在不同的地方。
- 当有重复文件时，一个目录项可由一个文件名和若干个索引结点指针组成。
- 在有重复文件时，如果一个文件拷贝被修改，则必须也同时修改其它几个文件拷贝，以保证各相应文件中数据的一致性。

文件名	i 结点
文件 1	17
文件 2	22
文件 3	12
文件 4	84

文件名	i 结点		
文件1	17	19	40
文件2	22	72	91
文件3	12	30	29
文件4	84	15	66

链接数的一致性检查（共享）

- 在 UNIX 系统中，每个目录项含有一个索引结点指针，用于指向文件的索引结点。
- 对于一个共享文件，其索引结点指针会在目录中出现多次。另一方面，在共享文件的索引结点中有一个链接计数 count，用于指向共享该文件的用户数。
- 在正常情况下，这两个数据应该是一致的，否则就出现了不一致性错误。
- 为了检查链接数一致性，要配置一张计数器表，为每个文件而不是为每个盘块建立一个表项，其中含有该索引结点指针的计数值。

链接数的一致性检查

- 检查时，从根目录开始查找，每当在目录中遇到该索引结点指针时，便在该计数器表中相应文件的表项上加 1。当把所有目录都检查完后，便可将每个表项中的计数值与该文件链接计数 count 值加以比较，如果两者一致，表示是正确的，否则便是产生了链接数据不一致的错误。
- 如果链接计数 count 值大于计数器表中相应的计数值，这种错误的后果是使一些已无用户需要的文件仍驻留在磁盘上，浪费了存储空间。解决的方法是用计数器表中的正确的计数值去为 count 重新赋值。

盘块号一致性检查

- 空闲盘块表 (链) 记录了所有尚未使用的空闲盘块的编号，文件分配表 FAT 则是用于记录已分配盘块的使用情况。
- 系统构造一张计数器表，每个盘块占一个表项。表中为每个盘块设立两个计数器。分别用作空闲盘块号计数器和数据盘块号计数器。
- 两组计数器中数据应该互补，否则说明发生了错误。

盘块号一致性检查

盘块号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
使用盘块	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
空闲盘块	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

正常盘块号

盘块号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
使用盘块	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
空闲盘块	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1

盘块号丢失

盘块号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
使用盘块	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
空闲盘块	0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1

空闲盘块记录错

盘块号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
使用盘块	1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
空闲盘块	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

使用盘块记录错

盘块号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
使用盘块	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0
空闲盘块	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

盘块号重复

本章作业

谢 谢

School of Computer & Information Engineering
Henan University
Kaifeng, Henan Province
475001
China