操作系统原理
第二十讲

马佳曼

jiaman.ma@nwpu.edu.cn

review

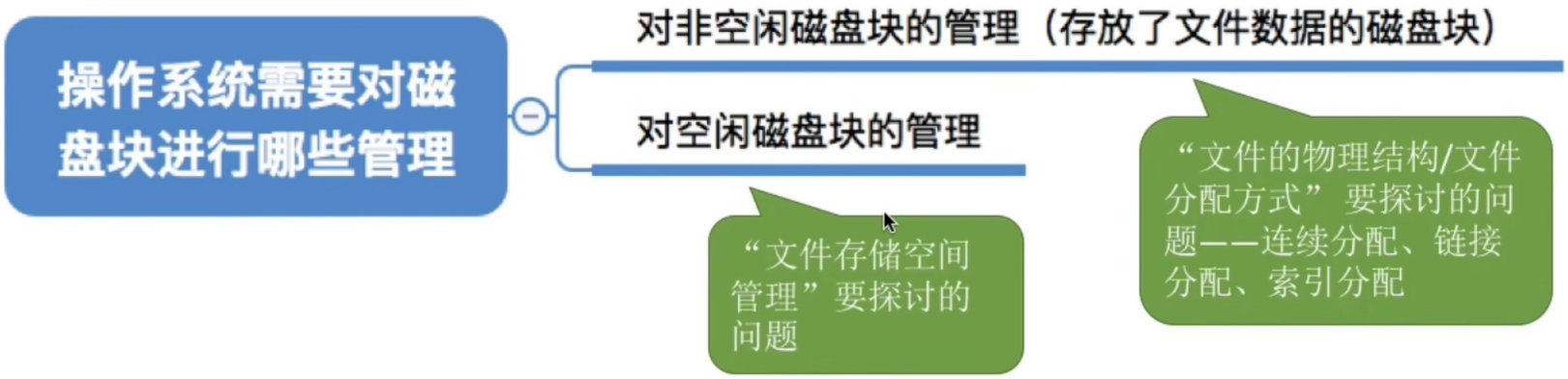
文件的目录

文件系统的使用

Today

文件存储空间的管理

文件的共享与保护



5.5 文件存储空间的管理

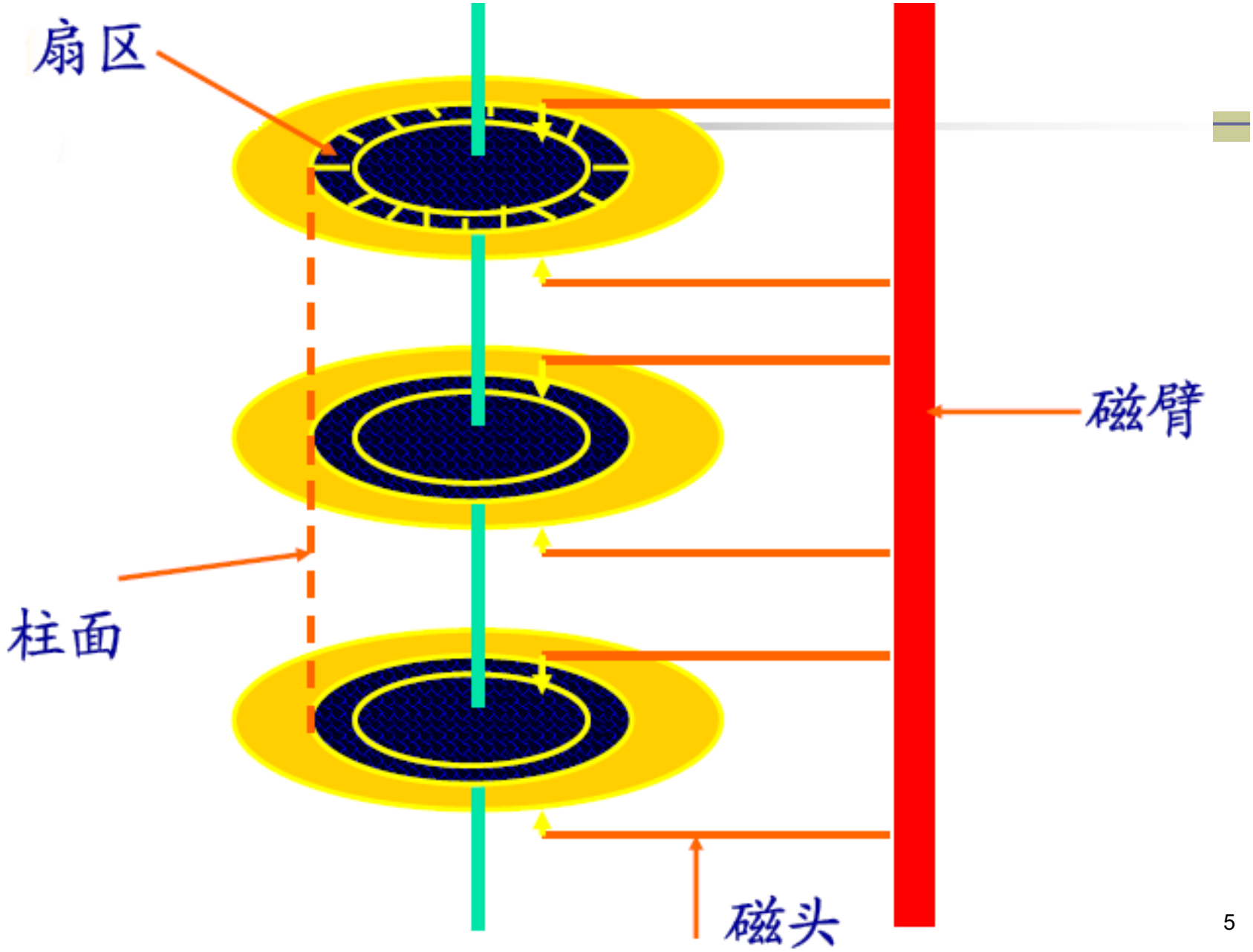
- 典型的存储介质

  磁盘（固态SSD）、磁带、光盘、U 盘等

- 物理块（块、簇）

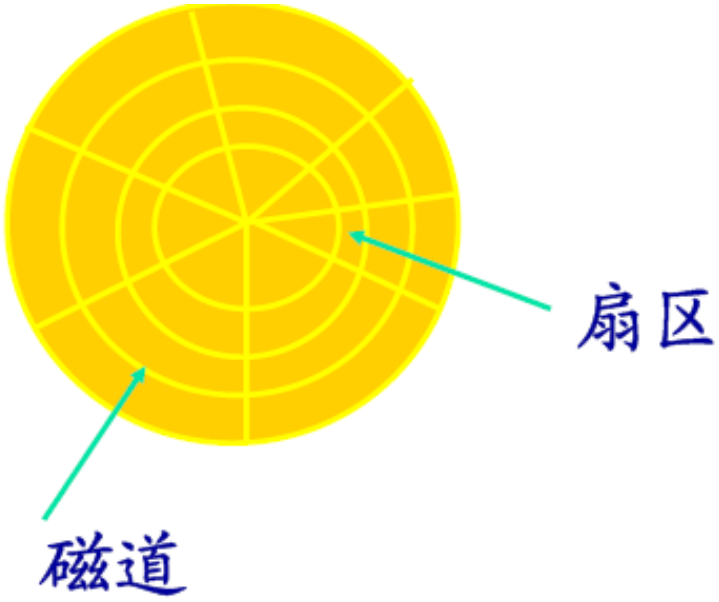
  信息存储、传输、分配的独立单位

  存储设备划分为大小相等的物理块，统一编号


扇区
磁臂
柱面
磁头

5.5 文件存储空间的管理

- 信息记录在磁道上，多个盘片，正反两面都用来记录信息，每面一个磁头
- 所有盘面中处于同一磁道号上的所有磁道组成一个柱面
- 物理地址形式：
  - 磁头号（盘面号）
  - 磁道号（柱面号）
  - 扇区号



访盘请求

- 磁盘系统由磁盘本身和驱动控制设备组成
- 直接（随机）存取设备：存取磁盘上任一物理块的时间不依赖于该物理块所处的位置
- 一次访盘请求：完成过程由三个动作组成：
  - 寻道（时间）：磁头移动定位到指定磁道
  - 旋转延迟（时间）：等待指定扇区从磁头下旋转经过
  - 数据传输（时间）：数据在磁盘与内存之间的实际传输

5.5.1 文件存储空间分配(file allocation)

新创建文件的存储空间（文件长度）分配方法：

- 预分配(preallocation)：创建时(这时已知文件长度)一次分配指定的存储空间，如文件复制时的目标文件。
- 动态分配(dynamic allocation)：需要存储空间时才分配（创建时无法确定文件长度），如写入数据到文件。

文件存储单位：簇（cluster）

- 簇的大小：大到能容纳整个文件，小到一个外存存储块；
  - 簇较大：提高I/O访问性能，减小管理开销；但簇内碎片浪费问题较严重；
  - 簇较小：簇内的碎片浪费较小，特别是大量小文件时有利；但存在簇编号空间不够的问题（如FAT12、16、32）；
- 文件卷容量与簇大小的关系
  - 文件卷容量越大，若簇的总数保持不变即簇编号所需位数保持不变，则簇越大。
  - 文件卷容量越大，若簇大小不变，则簇总数越多，相应簇编号所需位数越多。

文件存储分配数据结构

- 连续分配(contiguous)：只需记录第一个簇的位置，适用于预分配方法。可以通过紧缩(compact)将外存空闲空间合并成连续的区域。
- 链式分配(chained)：在每个簇中有指向下一个簇的指针。可以通过合并(consolidation)将一个文件的各个簇连续存放，以提高I/O访问性能。
- 索引分配(indexed)：文件的第一个簇中记录了该文件的其他簇的位置。可以每处存放一个簇或连续多个簇（只需在索引中记录连续簇的数目）。

5.5.3 外存空闲空间的管理 (free space management)

- 存储空间管理应解决的问题：
  - 如何登记空闲区的分布情况
  - 如何按需要给一个文件分配存储空间
  - 当某一文件或某一部分不再需要保留时，如何收回它所占用的存储空间
- 常用技术：
  - 空白文件目录（空闲表法）
  - 空白块链
  - 位示图

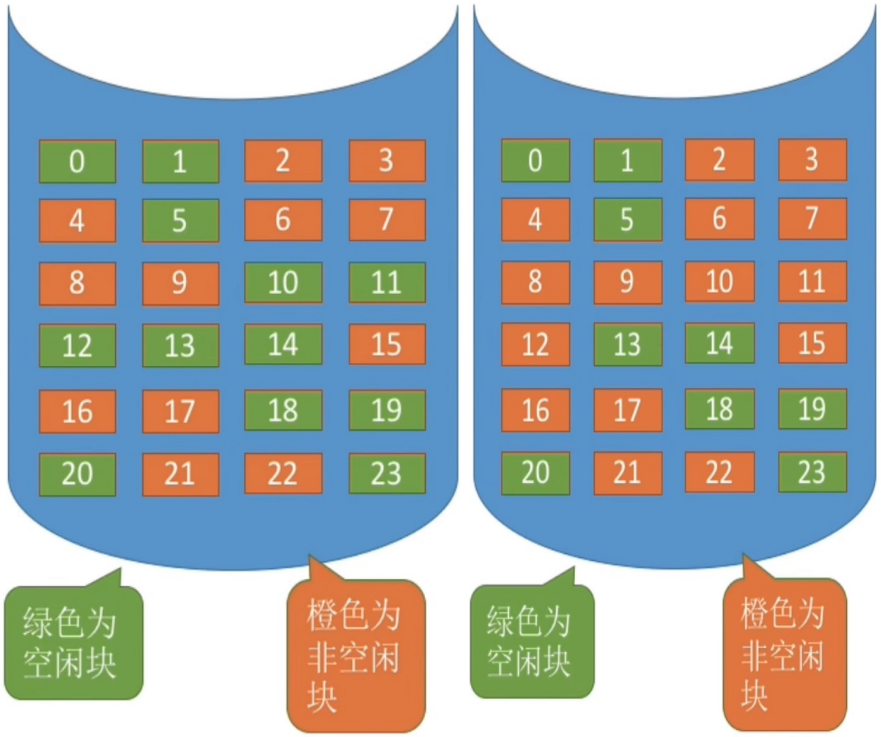
（1）空白文件目录（空闲表法）

- 辅存上的一片连续的空闲区，可视为一个空白文件，系统设置一张空白文件目录来记录辅存上所有空闲块的信息。每个表目存放一个空白文件的信息，包括该空白文件第一个空闲块号、 空闲块个数、该文件所有空闲块号等信息。

序号	第一个空闲块号	连续空闲块数	空闲块号
1	2	2	2，3
2	5	3	5，6，7
3	16	5	16，17，18，19，20

空白文件目录（空闲表法）

- 分配储存空间



第一个空闲盘块号	空闲盘块数
0	2
5	1
10	5
18	3
23	1

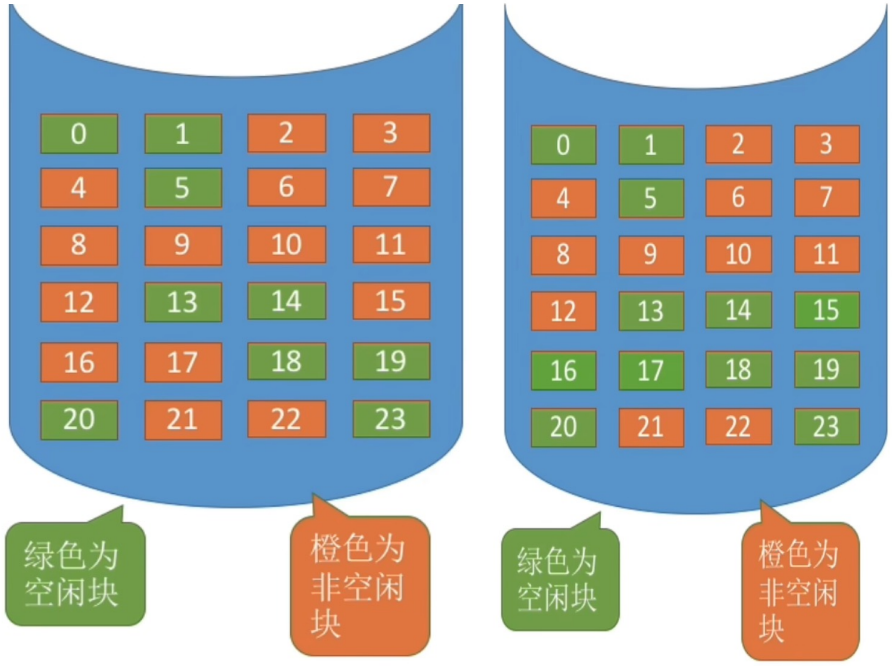
Eg：新创建的文件请求3个块，采用首次适应算法

空闲盘块表

如何分配磁盘块：与内存管理中的动态分区分配很类似，为一个文件分配连续的存储空间。同样可采用首次适应、最佳适应、最坏适应等算法来决定要为文件分配哪个区间。

空白文件目录（空闲表法）

- 回收储存空间



第一个空闲盘块号	空闲盘块数
0	2
5	1
13	8
23	1

空闲盘块表

情况② Eg：假设此时删除了某文件，系统回收了它占用的15、16、17号块

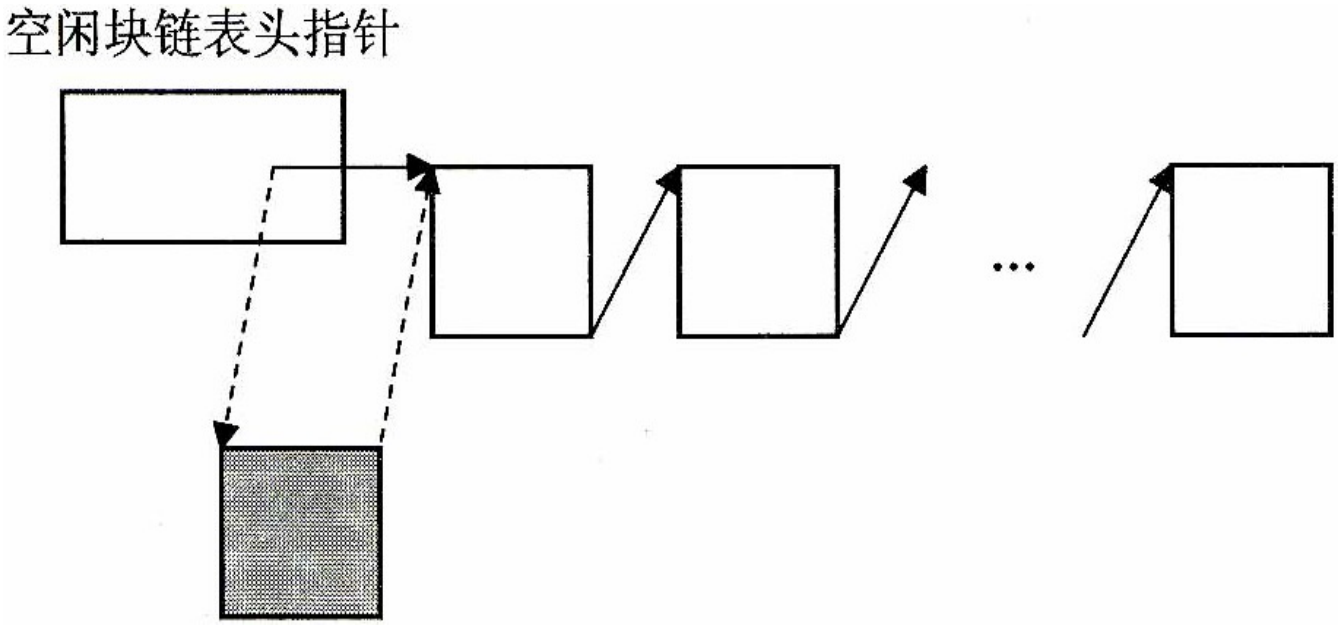
如何分配磁盘块：与内存管理中的动态分区分配很类似，为一个文件分配连续的存储空间。同样可采用首次适应、最佳适应、最坏适应等算法来决定要为文件分配哪个区间。

如何回收磁盘块：与内存管理中的动态分区分配很类似，当回收某个存储区时需要有四种情况——①回收区的前后都没有相邻空闲区；②回收区的前后都是空闲区；③回收区前面是空闲区；④回收区后面是空闲区。总之，回收时需要注意表项的合并问题。

- 这种方法适合于连续文件结构。但此方法有两个明显的缺点: □
  - (1) 如果文件太大, 那么在空白文件目录中将没有合适的空白文件能分配给它, 尽管这些空白文件的总和能满足需求。 □
  - (2) 经过多次分配和回收, 空白文件目录中的小空白文件越来越多, 很难分配出去, 形成碎片。

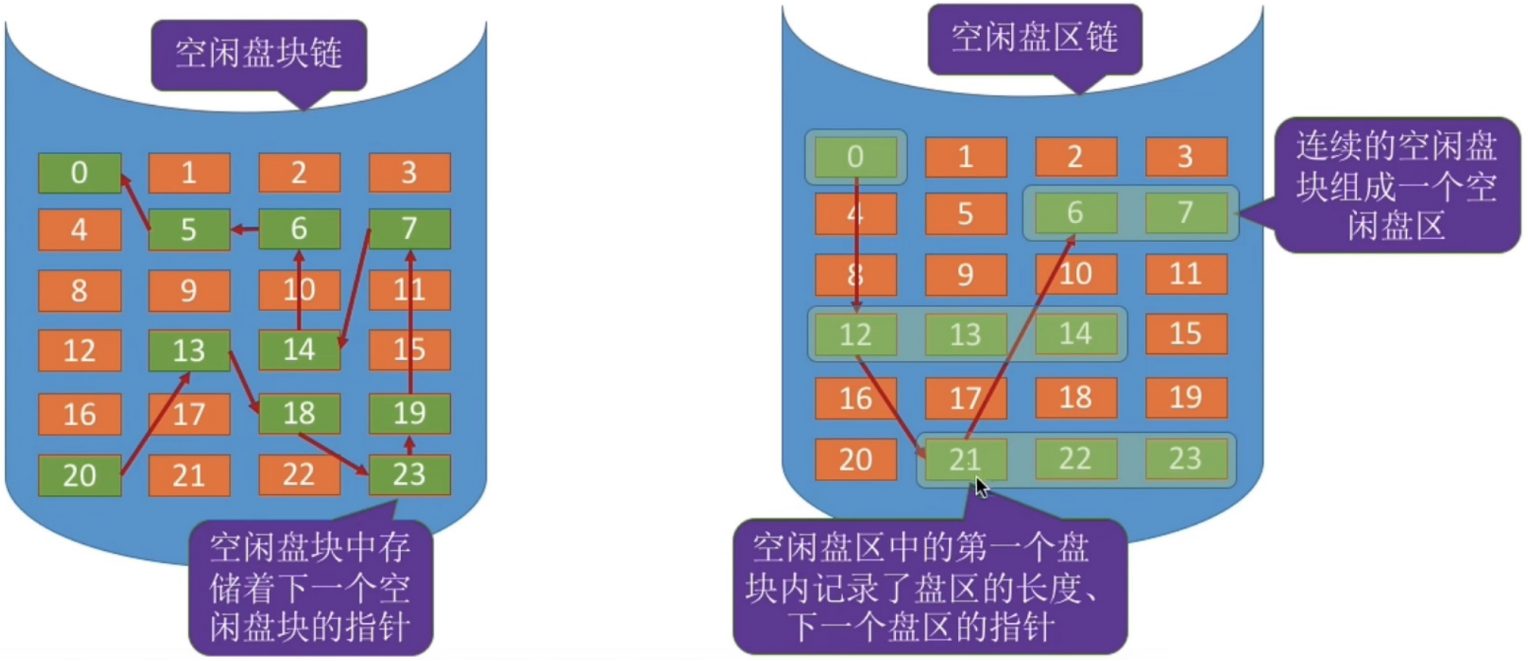
（2）空白块链

- 该方法把所有的空闲块链接在一起，形成一个空闲块链表。释放和分配空白快都可以在链首处进行。



空白块链（空闲链表法）

- (1)空闲盘块链。这是将磁盘上的所有空闲空间以盘块为单位拉成一条链。
- (2)空闲盘区链。这是将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。

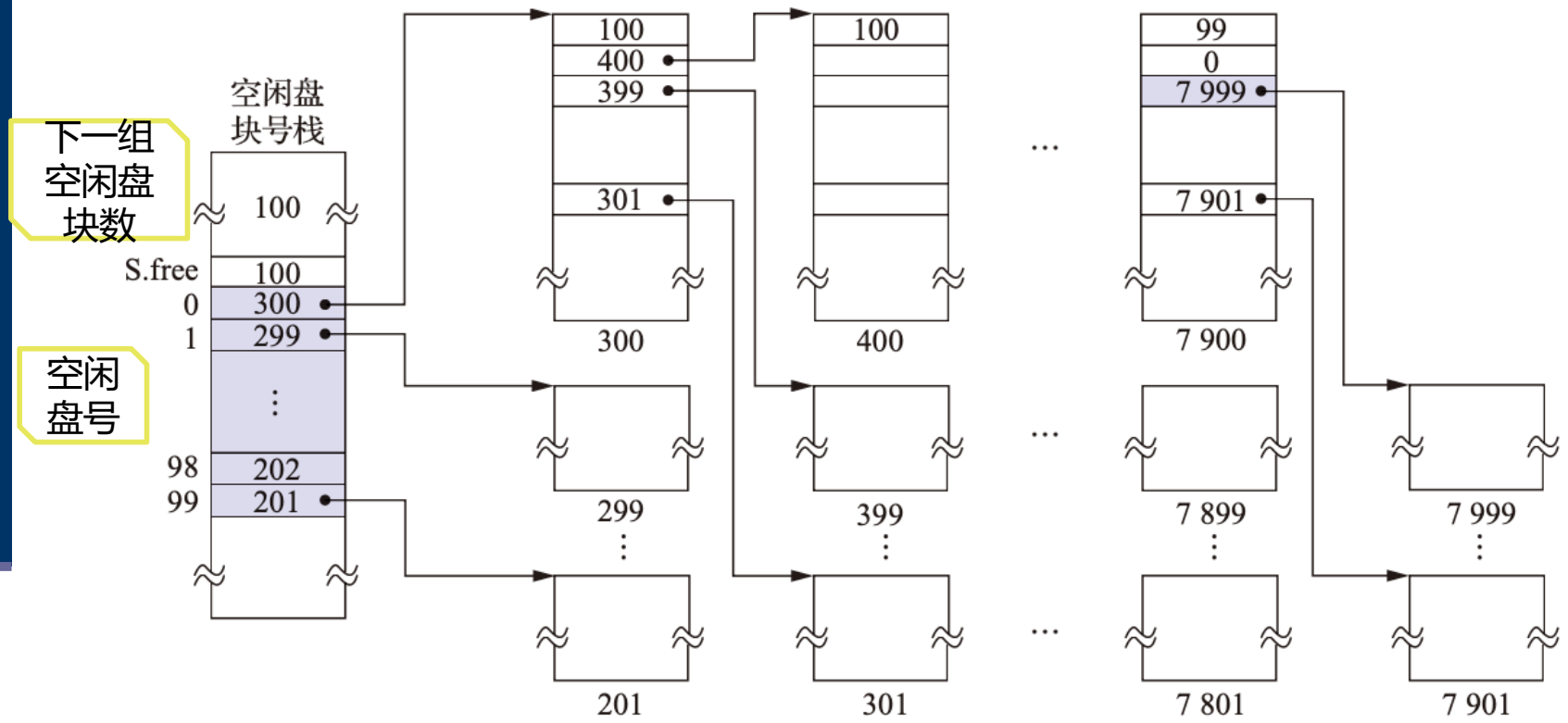


- 空闲链表法的优缺点如下:
  - （1） 可实现不连续分配。
  - （2） 节省了存储开销。
  - （3） 系统开销大。
  - （4） 对于大型文件系统, 空闲链将会太长。

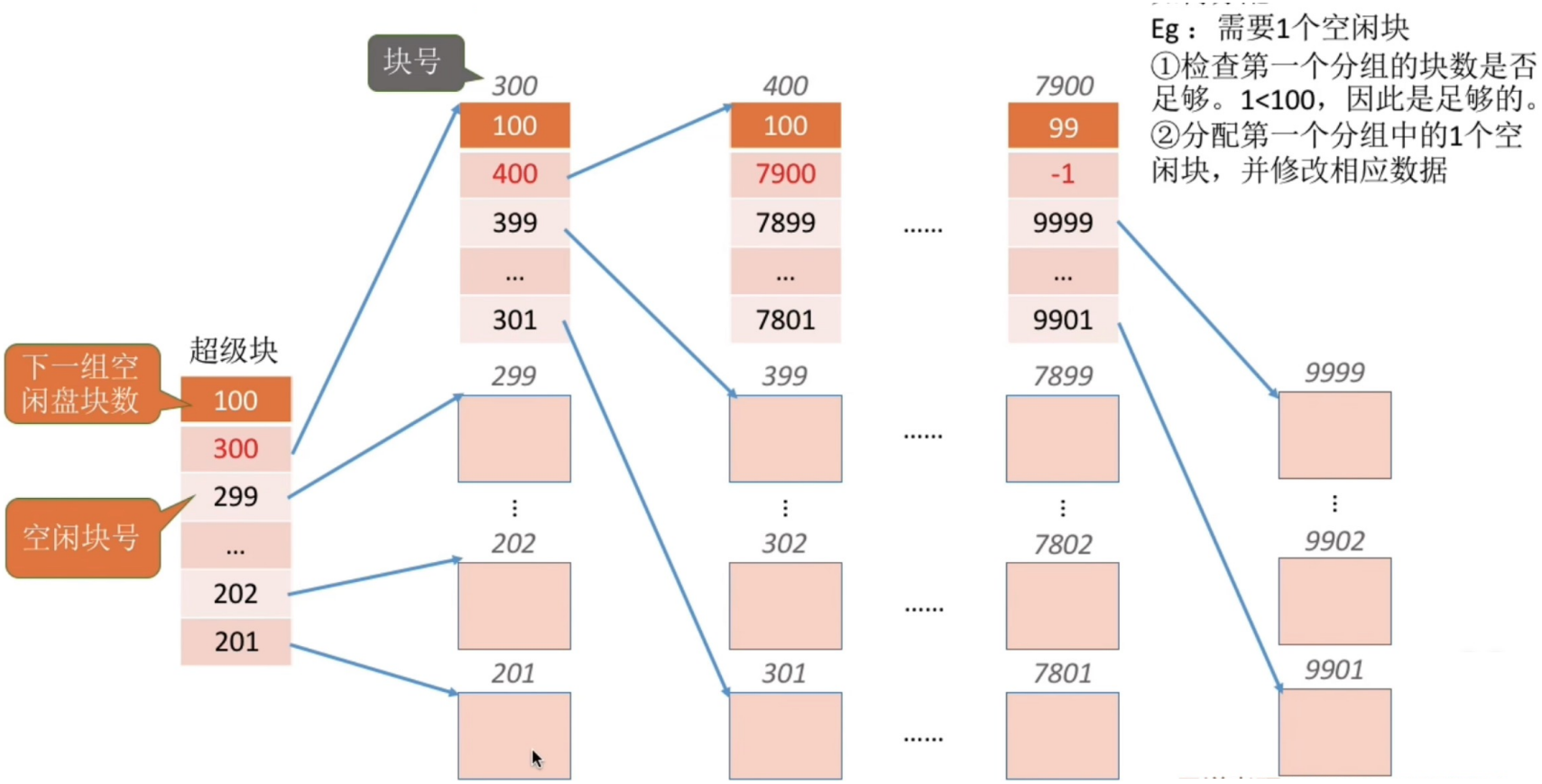
成组链接法

- UNIX使用空白块成组链接法
- 一个文件卷包括许多物理块：
  - 0#，引导块，用于引导操作系统
  - 1#，资源管理块（超级块），存放文件卷的资源管理信息
  - 2#开始，存放磁盘索引节点i-node块
  - 之后为一般的数据块
- 空闲盘块的分组
  - 按照从后往前的方法进行分组划分，所有空闲块按固定数量划分为若干组；
  - 每组的第一块用来存放前一组中各块的块号和块数，第一组的块数为N-1，最后一组可能不足N块；

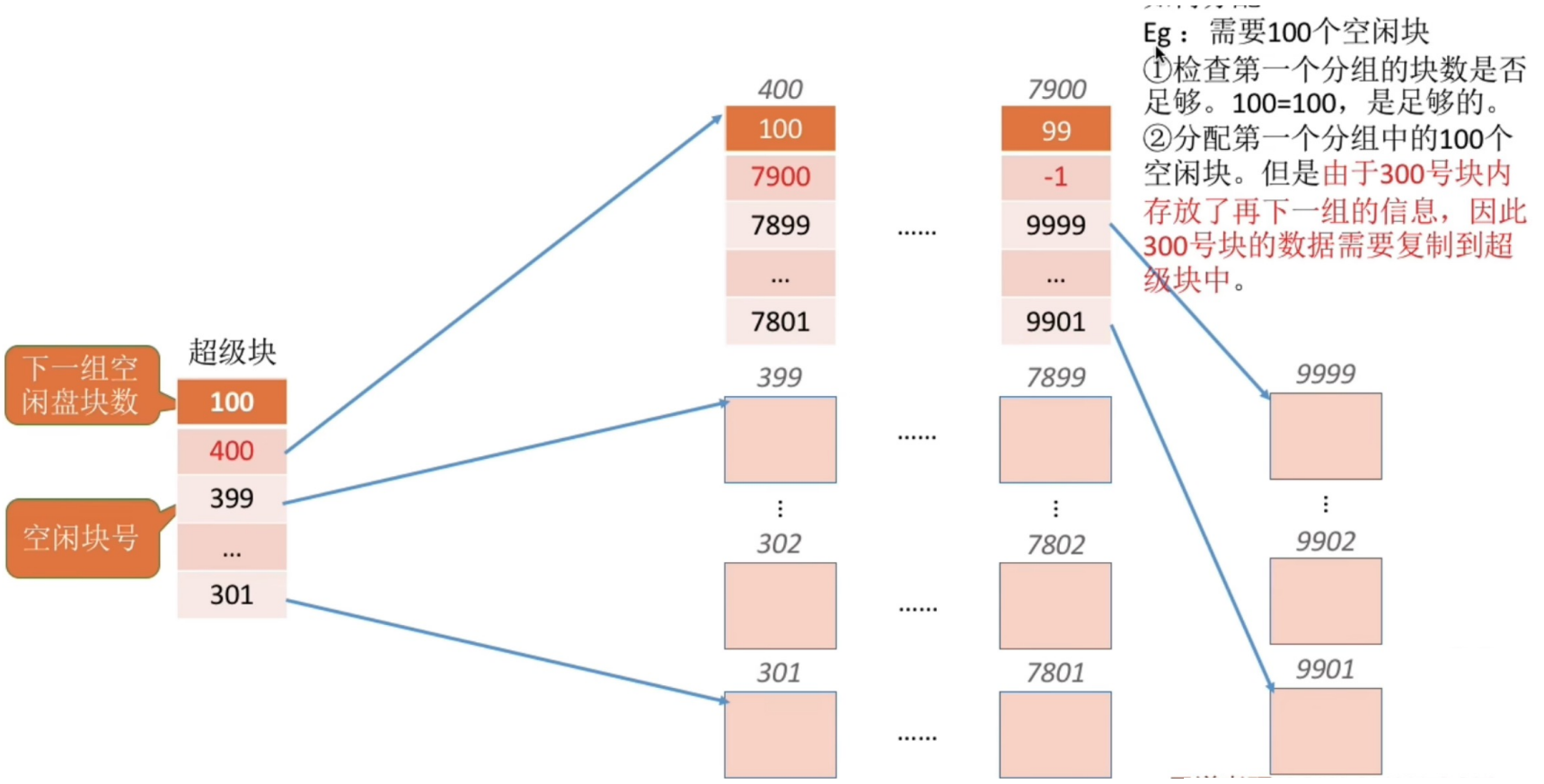
例：成组链接法


下一组
空闲盘
块数
空闲盘
块号栈
空闲
盘号
S.free
0
1
98
99
100
100
300
299
202
201
100
400
399
301
300
299
201
100
400
399
301
99
0
7 999
7 901
7 900
7 899
7 999
7 801
7 901

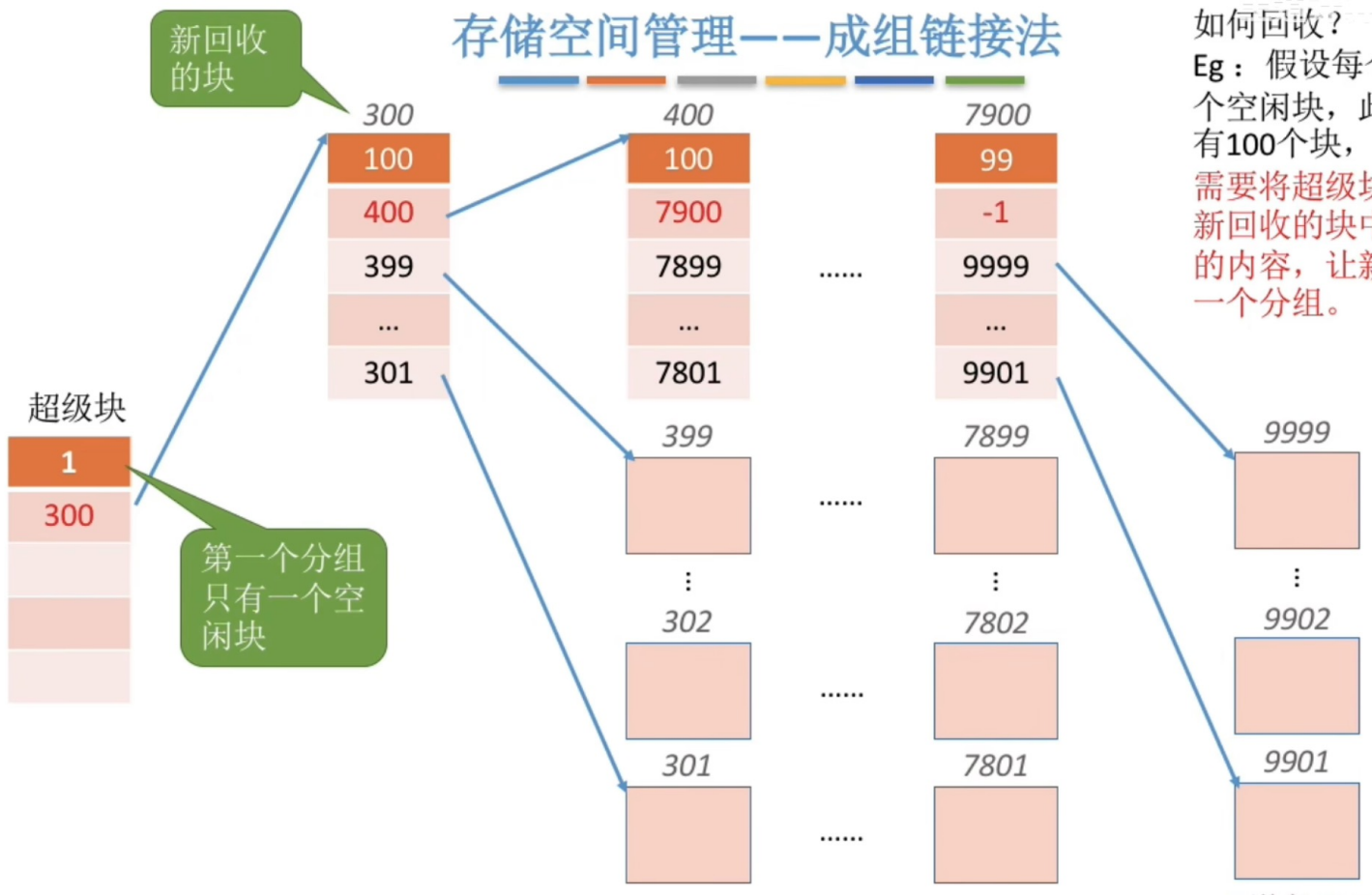
例：成组链接法


块号
超级块
下一组空闲盘块数
空闲块号
100
300
299
...
202
201
300
100
400
399
...
301
400
100
7900
7899
...
7801
7900
99
-1
9999
...
9901
299
399
7899
9999
202
302
7802
9902
201
301
7801
9901
.....
Eg： 需要1个空闲块
①检查第一个分组的块数是否足够。1<100，因此是足够的。
②分配第一个分组中的1个空闲块，并修改相应数据

例：成组链接法


超级块
下一组空闲盘块数
空闲块号
100
400
399
...
301
400
100
7900
7899
...
7801
399
302
301
7900
99
-1
9999
...
9901
7899
7802
7801
9999
9902
9901
.....
Eg： 需要100个空闲块
①检查第一个分组的块数是否足够。100=100，是足够的。
②分配第一个分组中的100个空闲块。但是由于300号块内存放了再下一组的信息，因此300号块的数据需要复制到超级块中。

例：成组链接法



如何回收？

Eg：假设每个分组最多为100个空闲块，此时第一个分组已有100个块，还要再回收一块。

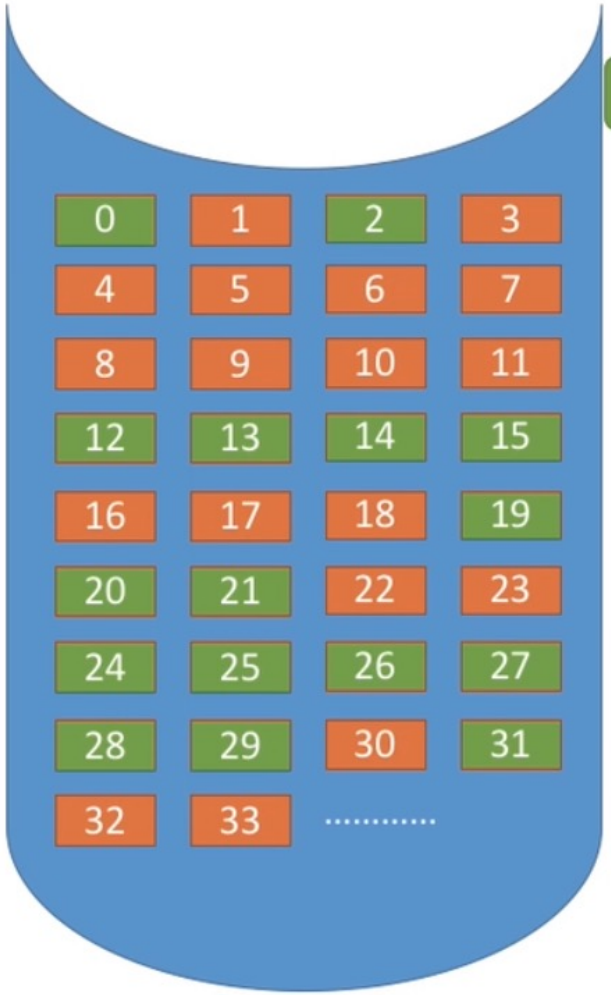
需要将超级块中的数据复制到新回收的块中，并修改超级块的内容，让新回收的块成为第一个分组。

- 空闲块的分配
  - 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
  - 若是最后一个盘块，将栈底盘块号所对应盘块的内容读入栈中，作为新的盘块号栈的内容，并把原栈底对应的盘块分配出去
- 空闲块的回收
  - 将回收盘块的盘块号记入空闲盘块号栈的顶部，并执行空闲盘块数加1操作
  - 当栈中空闲盘块号数目已达N时， 表示栈已满，便将现有栈中的N个盘块号， 记入新回收的盘块中，再将其盘块号作为新栈底

（3）位示图（Bit Map）

- 用一串二进制位反映磁盘空间分配使用情况，每个物理块对应一位，分配物理块为1，否则为0
- 申请物理块时，可以在位示图中查找为0的位，返回对应物理块号；归还时；将对应位转置0
- 描述能力强，适合各种物理结构

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																



位号

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

位示图：每个二进制位对应一个盘块。在本例中，“0”代表盘块空闲，“1”代表盘块已分配。位示图一般用连续的“字”来表示，如本例中一个字的字长是16位，字中的每一位对应一个盘块。因此可以用（字号，位号）对应一个盘块号。当然有的题目中也描述为（行号，列号）

重要重要重要：要能自己推出盘块号与（字号, 位号）相互转换的公式。

注意题目条件：盘块号、字号、位号到底是从0开始还是从1开始

如本例中盘块号、字号、位号从0开始，若n表示字长，则...

(字号, 位号)=(i, j) 的二进制位对应的 盘块号 b = ni + j

磁盘的分配（从 1 开始）

- （1） 顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位(“0”表示空闲)。
- （2） 将所找到的一个或一组二进制位， 转换成与之相应的盘块号。假定找到的其值为 “0”的二进制位，位于位示的第i行、第j列，则其相应的盘块号应按下式计算：

$$b=n(i-1)+j$$

式中， n代表每行的位数。

- （3） 修改位示图， 令map [i,j] =1。

磁盘的回收

- （1）将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：

  i=(b-1)DIV n+1

  j=(b-1)MOD n+1

- （2）修改位示图。 令map [i,j] =0

例：1.2M磁盘，以512个字节为一块，试问位示图最大为多少字节？

$$\frac{1.2M}{512}\times\frac{1}{8}=300\text{个字节}$$

有一计算机系统利用图 1-9-2 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，则请回答下列问题：

(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，则应如何处理？

i/j	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	1	0	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
…																

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i=2，列号j=2;

2. 2 计算出该位所对应的盘块号：

$b=i\times16+j+1=2\times16+2+1=35$;

3. 修改位示图，令map[2,2]=1，并将对应的盘块35分配给文件。按照同样的方式，可找到第3行、第6列的值为0的位，将其转换为盘块号55;将位的值修改为1，并将55号盘块分配给文件。

(2)释放磁盘的第300块时，应进行如下处理:

计算出磁盘第300块所对应的二进制位的行号i和列号j，即

i=(300−1)DIV 16=18，j=(300−1)MOD 16=11;

修改位示图，令map[18,11]=0，表示对应的盘块为空闲块。

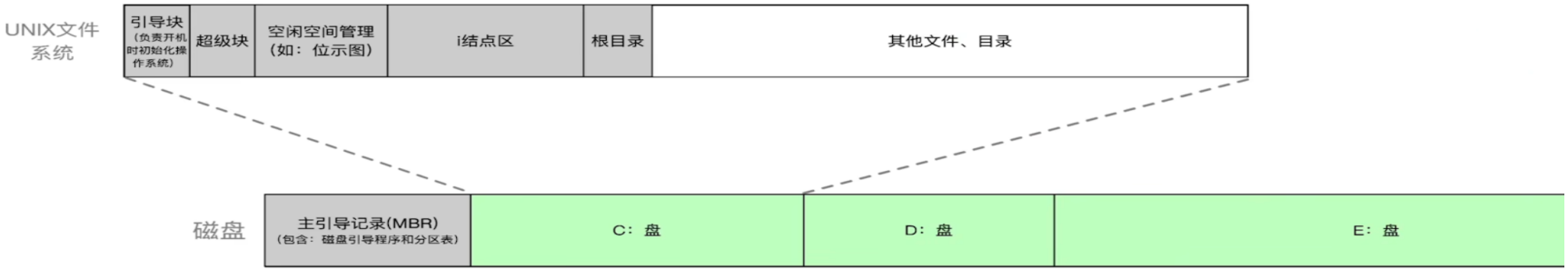
5.5.4 文件卷

- 磁盘分区(partition)：通常把一个物理磁盘的存储空间划分为几个相互独立的部分，称为“分区”。
- 文件卷(volume)：或称为"逻辑驱动器(logical drive)"。在同一个文件卷中使用同一份管理数据进行文件分配和外存空闲空间管理，而在不同的文件卷中使用相互独立的管理数据。
  - 一个文件不能分散存放在多个文件卷中，其最大长度不超过所在文件卷的容量。
  - 通常一个文件卷只能存放在一个物理外设上（并不绝对），如一个磁盘分区或一盘磁带。

- 格式化(format)：在一个文件卷上建立文件系统，即：
  - 建立并初始化用于进行文件分配和外存空闲空间管理的管理数据。
  - 通常，进行格式化操作使得一个文件卷上原有的文件都被删除。
- 磁盘交叉存储(disk interleaving)：将一个文件卷的存储块依次分散在多个磁盘上。如4个磁盘，则磁盘0上是文件卷块0, 4, 8, …，磁盘1上是文件卷块1, 5, 9, …。
  - 优点：提高I/O效率。
  - 需要相应硬件设备

物理格式化&逻辑格式化

物理格式化，即低级格式化——划分扇区，检测坏扇区，并用备用扇区替换坏扇区



逻辑格式化后，磁盘分区（分卷 Volume），完成各分区的文件系统初始化
注：逻辑格式化后，灰色部分就有实际数据了，白色部分还没有数据

5.6 文件的共享与保护

- 文件共享的目的：
  - 节省存储空间
  - 进程间通过文件交换信息
- 共享和保护是一个问题的两个方面
  - 共享：一个或一部分文件由事先规定的某些用户共同使用
  - 保护：文件不得被未经文件主授权的任何用户使用

文件共享的方式

- 同名共享：
  - 各个用户使用同一个文件名（包括其路径）来访问某一文件。
- 异名共享：
  - 各个用户使用各自不同的文件名来访问某个文件。
  - 异名共享所采用的方法称为文件的勾链：

    1）基于索引点的共享方法

    容许目录项链接到目录树中任一节点上

    2）基于符号链的共享方法

    只容许链接到数据文件的叶子节点上

实例

- 各实际OS是否提供链接技术
  - DOS ×
  - Windows √（快捷方式）
  - Unix √（硬链接/软链接）
- 用户程序cc在运行时要用到目录/lib下的文件mad，但后来包括mad在内的一些文件被整理到/usr/lib下，并改名为mad1，为使cc正常运行，应使用
- ln /usr/lib/mad1 /lib/mad

文件的打开结构共享

- 三部分组成：
  - 进程打开文件表
  - 系统打开文件表
  - 内存inode
- 父子进程打开文件的共享
- 同名或异名打开文件的共享

文件的保护

- 对拥有权限的用户，应该让其进行相应操作，否则应禁止。
  - 对用户进行分类
  - 对访问权限分类
  - 用访问控制矩阵实现文件保护
  - 存取控制表实现文件保护
  - 用户权限表实现文件保护
  - 用口令实现文件保护

对用户进行分类

- 按用户对文件访问权力的差别把用户分成几类，然后对每个文件规定各类用户的存取权限。通常将用户分成三类：
  - 文件主
  - 文件主的同组用户或合作用户
  - 其它用户

对访问权限分类

- 对文件的访问系统首先要检查访问权限，只允许合法的用户访问。文件的存取权限一般有以下几种：
  - 仅允许执行 (E)。
  - 仅允许读 (R)。
  - 仅允许写 (W)
  - 仅允许在文件尾写 (A)
  - 仅允许对文件进行修改（U）
  - 允许改变文件的存取枚限（C）
  - 允许取消文件（D）
- 这几种权限可进行适当的组合。

用户权限表实现文件保护

用户 \ 文件	F1	F2	…
zhang	RW	R	…

用访问控制矩阵实现文件保护

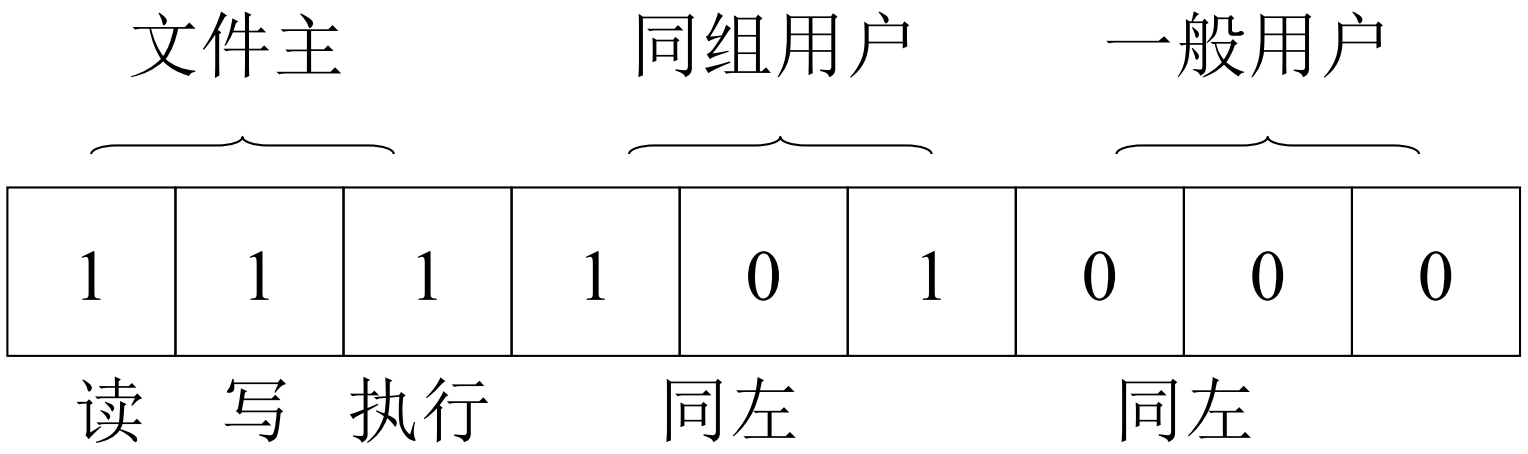
- 一维代表所有用户，一维代表系统中的所有文件。
- 优点：一目了然
- 缺点：矩阵往往过大。

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

用存取控制表实现文件保护

文件名：F1	
文件主	RWE
同组成员	RW
其他成员	R

例：UNIX文件描述符说明三类用户的名字。



UNIX的文件访问控制

采用文件的二级存取控制

审查用户的身份、审查操作的合法性

第一级：对访问者的识别

对用户分类：

- ✓文件主（owner）
- ✓文件主的同组用户（group）
- ✓其他用户（other）

第二级：对操作权限的识别

对操作分类：

- ✓读操作（r）
- ✓写操作（w）
- ✓执行操作（x）

不能执行任何操作（-）

例子：rwx rwx rwx

chmod 711 file1 或 chmod 755 file2

用口令实现文件保护

- 存取控制表、用户权限表都将占据大量存储空间，可采用另一种较简单的方法：口令。
- 用户为自己的每一个文件设置一个口令，存放在文件的FCB中。 任何用户要存取该文件，都必须提供和FCB中一致的口令，才有权存取。
- 优点：简便，空间开销小，验证开销小
- 缺点：
  - 保护级别少（可访问和不可访问）
  - 保密性差，存在系统内部，不安全
  - 不易改变存取控制权限。

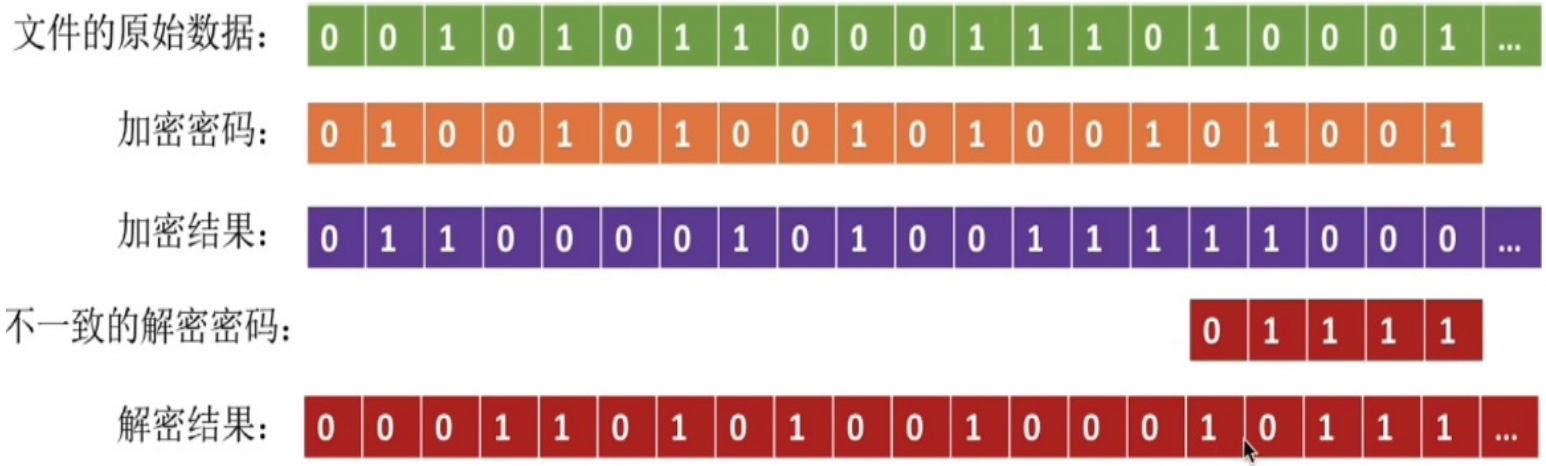
用密钥（Encryption）实现文件保护

- 在文件建立保存时，加密程序根据用户提供的代码键对文件进行编码加密，在读取文件时，用户提供相同的代码键，解密程序根据该代码键对加密文件进行译码解密，恢复为源文件。
- 只有知道代码键的用户才能正确访问文件，且代码键不存放在系统中，故该方法保密性很强。但耗费大量编码、译码时间，系统开销大而且降低了访问速度。
- 一般可将几种安全控制手段综合使用。

加密保护

使用某个“密码”对文件进行加密，在访问文件时需要提供正确的“密码”才能对文件进行正确的解密。

Eg：一个最简单的加密算法——异或加密
假设用于加密/解密的“密码”为“01001”



优点：保密性强，不需要在系统中存储“密码”
缺点：编码/译码，或者说加密/解密要花费一定时间。

5.6 文件系统的管理

文件系统的可靠性

- 可靠性：
  抵御和预防各种物理性破坏和人为性破坏的能力

- 坏块问题
- 备份
- 通过转储操作，形成文件或文件系统的多个副本

文件系统备份

全量转储：
定期将所有文件拷贝到后援存储器 **增量转储：**
只转储修改过的文件，即两次备份之间的修改，减少系统开销

物理转储：
从磁盘第0块开始，将所有磁盘块按序输出到磁带 **逻辑转储：**
从一个或几个指定目录开始，递归地转储自给定日期后所有更改的文件和目录