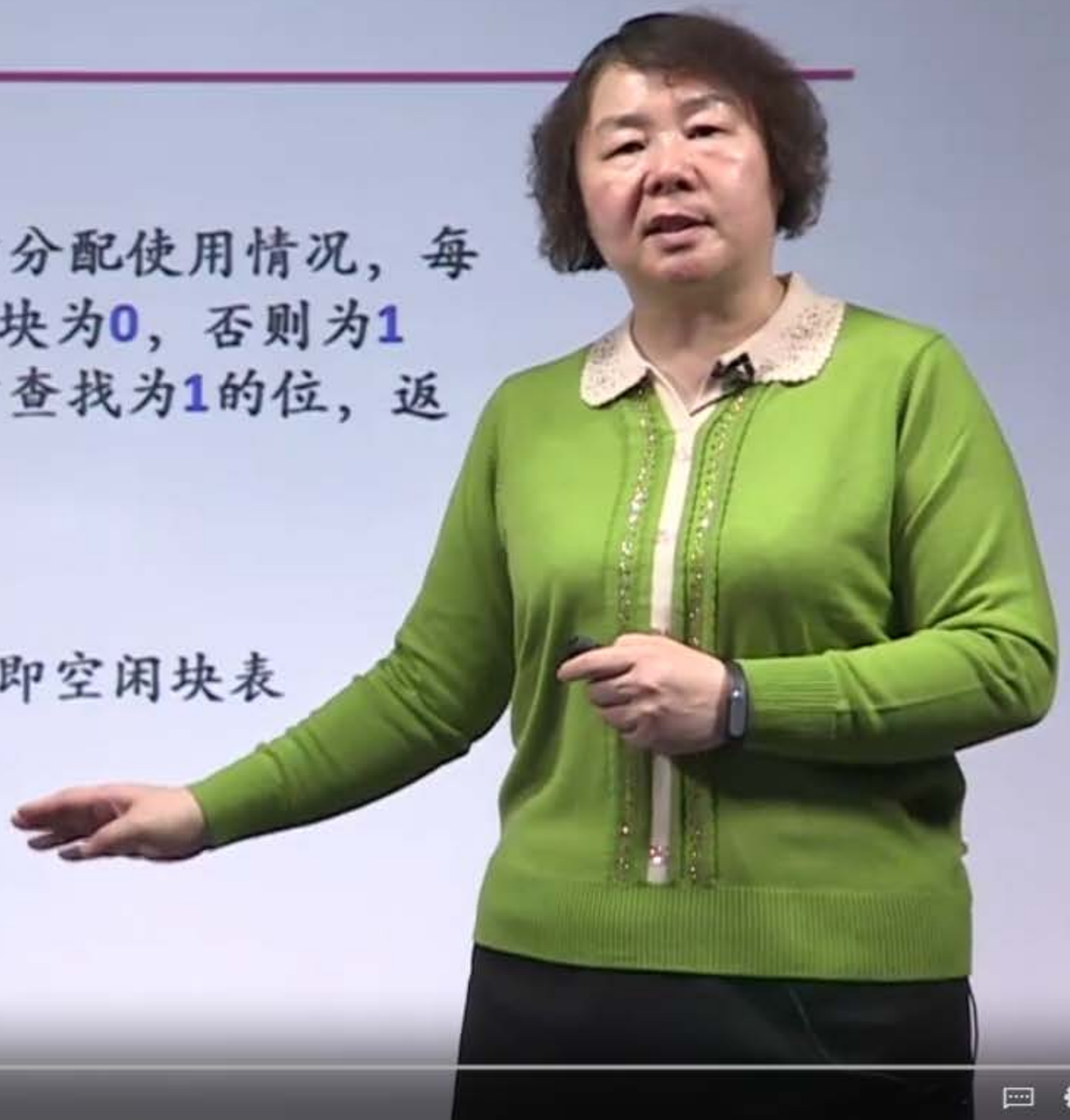
有关数据结构

- 位图
  - 用一串二进制位反映磁盘空间中分配使用情况，每个物理块对应一位，分配的物理块为0，否则为1
  - 申请物理块时，可以在位示图中查找为1的位，返回对应物理块号
  - 归还时，将对应位转置1
- 空闲块表
  - 将所有空闲块记录在一个表中，即空闲块表
  - 主要两项内容：起始块号，块数
- 空闲块链表
  - 把所有空闲块链成一个链
  - 扩展：成组链接法 ✓



磁盘地址与块号的转换

已知块号，则磁盘地址：

柱面号＝[块号/（磁头数×扇区数）]

磁头号＝[（块号mod （磁头数×扇区数））/扇区数]

扇区号＝（块号mod （磁头数×扇区数））mod 扇区数

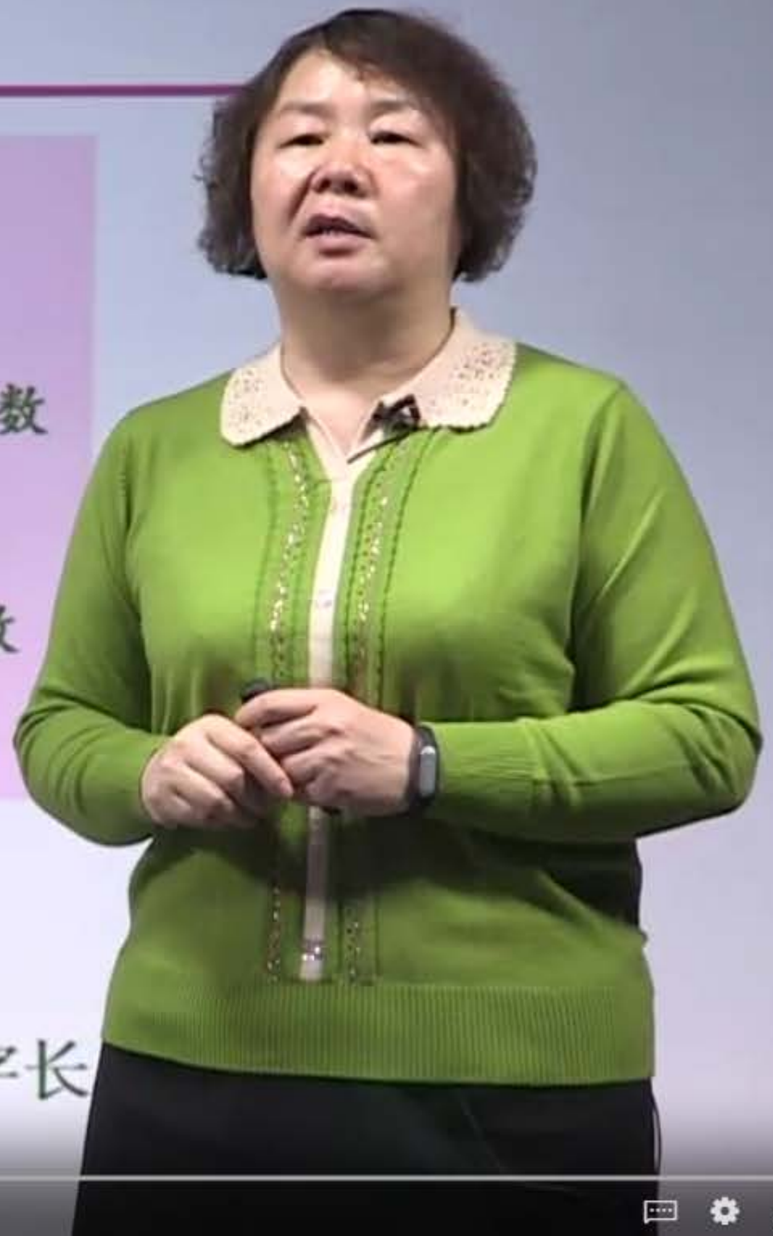
已知磁盘地址：

块号＝柱面号×（磁头数×扇区数）＋磁头号×扇区数＋扇区号

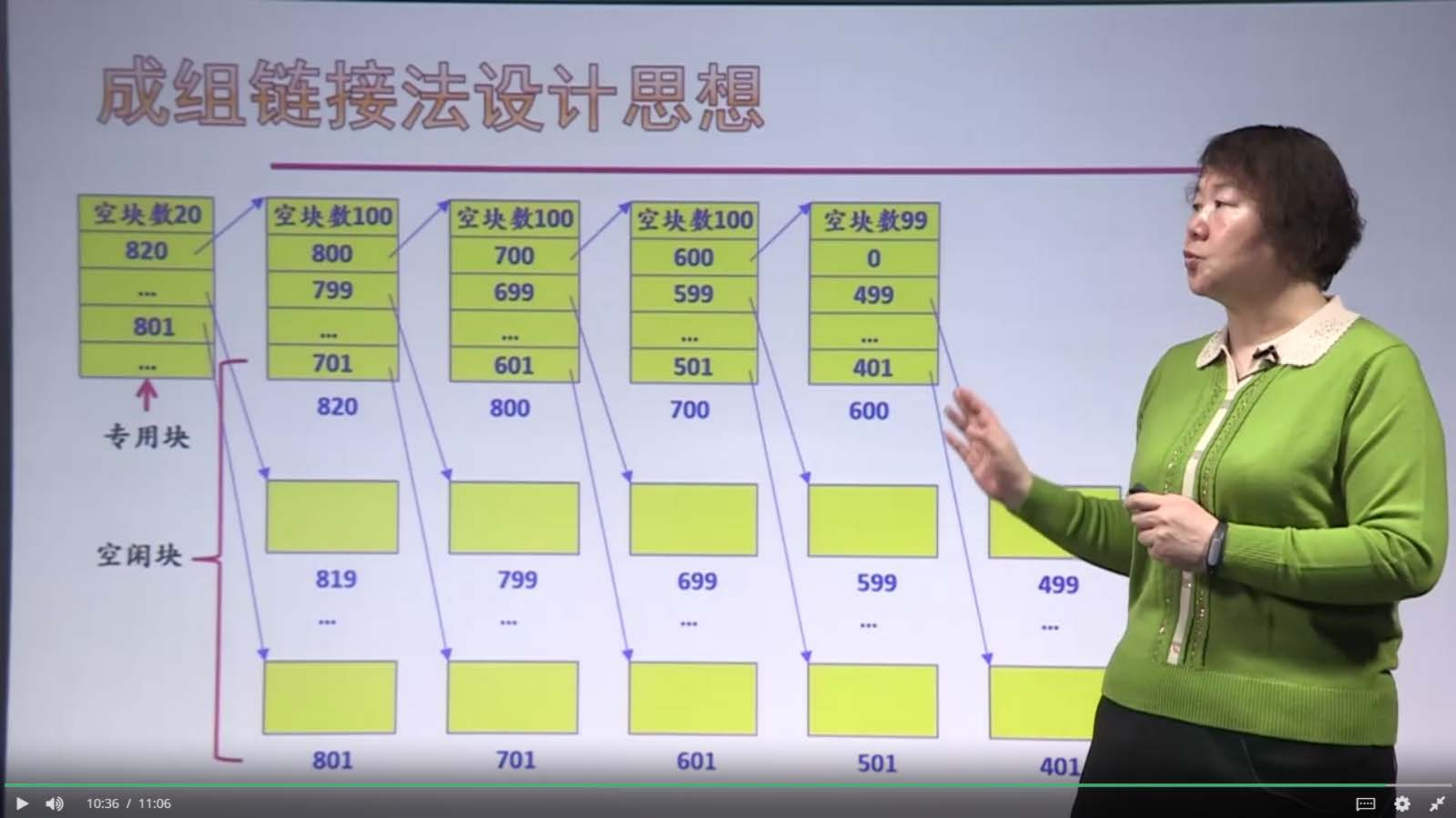
位图计算公式：

已知字号i、位号j：块号＝i×字长＋j

已知块号：字号＝[块号/字长]　位号＝块号 mod 字长



成组链接法设计思想


空块数20
820
...
801
...
专用块
空闲块
空块数100
800
799
...
701
820
819
...
801
空块数100
700
699
...
601
800
799
...
701
空块数100
600
599
...
501
700
699
...
601
空块数99
0
499
...
401
600
599
...
501
499
...
401

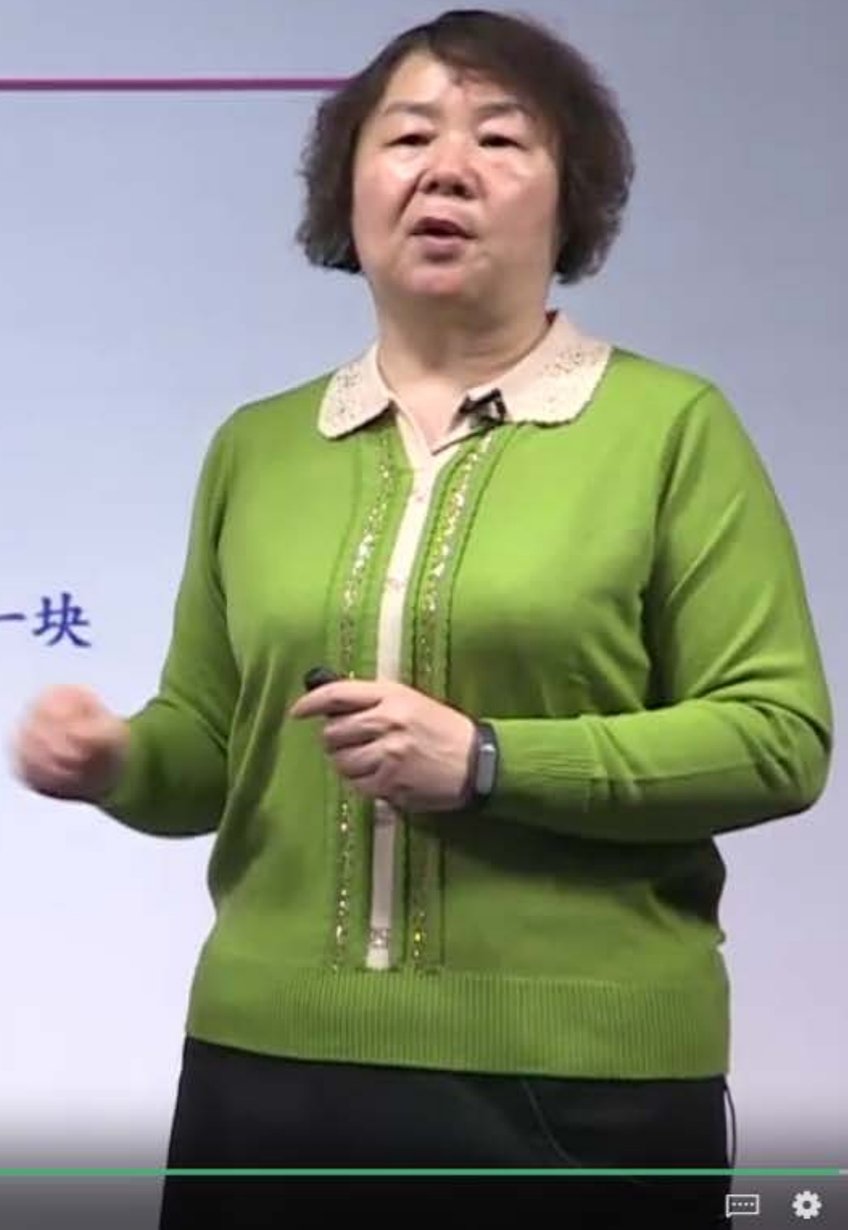
成组链接法—分配算法

分配一个空闲块

查L单元（空闲块数）：

- 当空闲块数>1 i＝L＋空闲块数;
  从i单元得到一个空闲块号;
  把该块分配给申请者;
  空闲块数减1;
- 当空闲块数＝1 取出L＋1单元内容（一组的第一块块号或0）；
  其值＝0 无空闲块，申请者等待
  其值不等于零，把该块内容复制到专用块;
  该块分配给申请者;

把专用块内容读到内存L开始的区域



成组链接法—回收算法

归还一块：

查L单元的空闲块数；

- 当空闲块数<100 空闲块数加1；

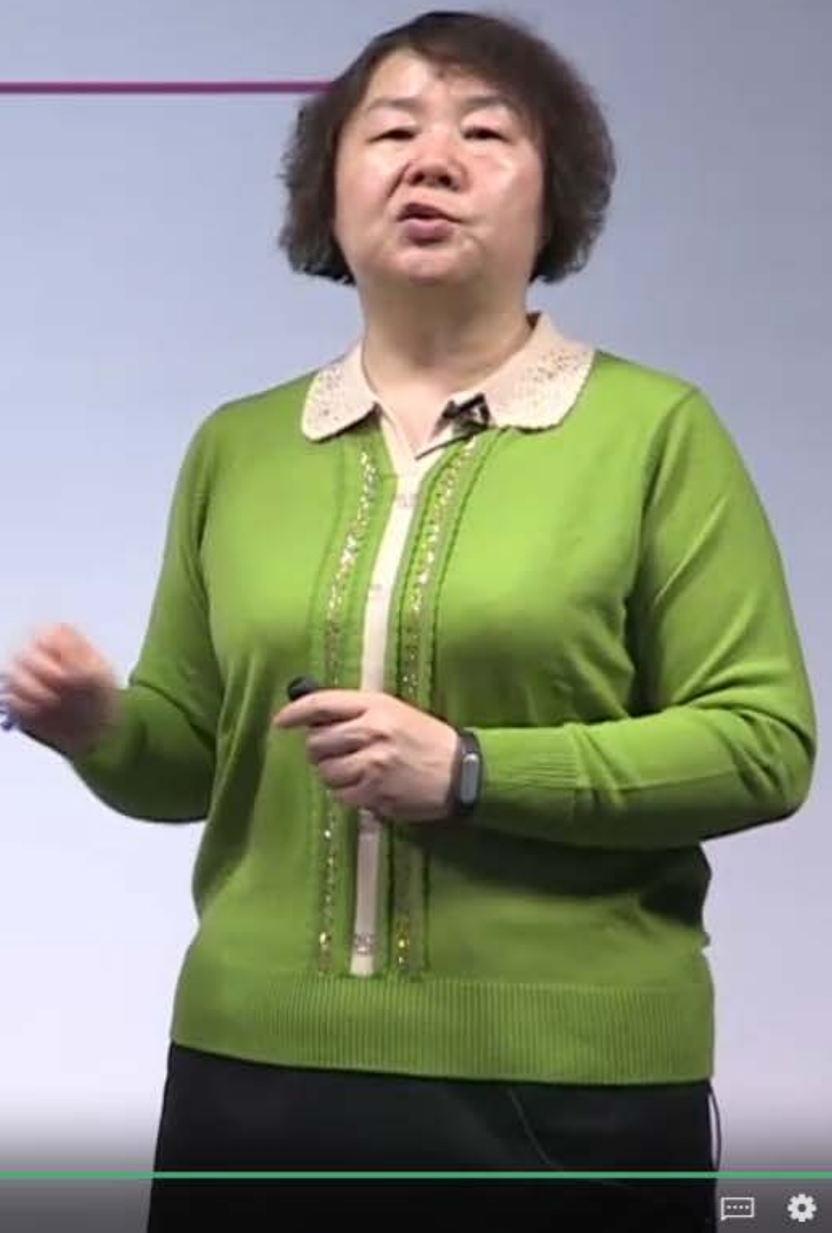
  j:＝ L＋空闲块数；

  归还块号填入j单元。

- 当空闲块数＝100，则把内存中登记的信息写
  归还块中；

  把归还块号填入L＋1单元；

  将L单元置成1。



0:00 下面我们介绍一下磁盘空间的管理。 首先我们 看一下，磁盘空间管理有哪些可采用的数据结构？ 位图、空闲块表、 空闲块链表 位图呢，实际上就是用一串 二进制位来反映磁盘空间的分配回收情况 每一个物理块对应了 位图当中的一位。 如果这个物理块是分配出去的，那么这一位呢就设置为 0 如果这个物理块还是空闲的，就设置为 1 因此当一个文件需要申请 物理块时，那么可以在位图中来查找 唯一的位，然后返回给对应的物理块号 在回收一个物理块时呢，就将对应的位 设置成 1 ，那么空闲块表呢 主要是把所有的空闲块记录在这张表里头 那么主要记的是两项信息。 一项呢是 这片空闲块的起始块号 以及连续有多少个空闲块 那么空闲块链表呢 最简单的组织方式就是把所有的空闲块，链成一个 链。 当然了，每一个物理块，那么如果 它的下一个空闲物理块，那么拿这个物理块当中的 因为它是空闲的，会用其中一部分的字节当作指针，指向下一个空闲的物理块 那这样的话，这个链就会很大，啊 系统就会变得支离破碎，因此呢 通常呢，系统会用一些改进的方法 那么待会呢，我会去介绍一下 Unix 的一种改进 成组链接法 这里给出了一些磁盘地址和块号的一个换算关系 如果我们知道了块号，比如说我们说一个啊盘 有 10000 块那么 0 到9999 块 那我们已经知道了块号，我可以算出它的磁盘地址 磁盘地址是由柱面号，磁头号和扇区号来组成的 如果我知道的一个磁盘地址，我也可以换算出它的 块号，那这就是一个换算的公式。 那么在位图当中 也可以通过计算，得到对应的块号。比如说我已经知道了字号 和位号，因为我们的位图是由若干个字组成，那么我们 知道了字号和位号的话呢，我们可以通过这个公式算出来它的什么呀，块号 我们知道了块号之后呢，我们也可以把它分解成位号和这个字号 这样的话呢，我们就可以在位图当中某一个位，把它设置 0 或者 1 这就是磁盘地址和块号的一个换算关系 那么下面我们介绍一下，成组链接法 那它的设计思想呢是这样的。 首先呢，我们来看一下 这张图里头，最左边有一个磁盘块 物理块，我们把它称之为叫专用块 那么也就是说，这个块里头的内容是一个有用的信息 然后我们可以看到，这一片 都是空闲块，啊，在我们这有一个啊 大括号，就表示这些都是空闲块 这些空闲块呢，我们把它分成若干组 这里头呢我们就典型的啊，我们说 100 块是一组 那我们看一下，比如说这是第一组 这是第二组，第三组等等等等 那么我们可以看到，比如说最后这一组啊，大家可能看一下 最后这一组，它有 99 空闲块，它少 1 个空闲块，这是我们一个刻意设计 然后，倒数第二组 有 100 个空闲块，那我们为了说明它是 100 个，所以我们的括号刻意标志成什么 501 啊，502 到 599 这样的话呢，到 600 正好是 100

块 但实际的系统，可能不是这样一个非常规整的连续的啊数字 然后我们依次啊分组，那我们知道最后这一组或者说第一组 它应该不足 100 块是可能的，所以我们这里头 正好是这一组是二十块，也就是第一组只有二十个空闲块 首先我们把这个布局看清楚 那我们来看一下，这二十个空闲块 啊第一组的二十个空闲块，就有二十个括号 这二十个括号都保存在哪里呢？我们可以看到保存在了专用块里头 大家可以看一下，在专用块里头有若干个字段，第一个字段呢，其实是就存放了 它的第一组的空闲块的块数，所以就是说空闲块块数现在是二十 接着的各个字段，就把刚才我们所说的 二十个空闲块的块号，记录在了每一个字段里头 那么按照这个顺序记录。 那么每次要想 一个文件想获取一个新的空闲块的话呢 就从专用块来挑选空闲块 挑选的时候呢，是从下往上挑选。 也就是在这里 头就示意就说，如果先分配的话，一定是 801 这一块 然后接着分配 802 这一块，最后才分配 820 这一块。 就是这样一个关系 那我们可以看一下说，当一个文件 它说我现在需要用十块，那么怎么做呢？实际上当文件 系统，当系统启动了之后呢，这个专用块呢，就把它读入内存了 读入内存以后呢，我们可以看到那么 可以判断第一组里头还剩下多少个空闲块？刚刚我们说一个文件需要十块 现在有二十块，那么可以满足这个文件的要求，因此呢就把这个二十 呢 减十，那么还空余啊，空闲、剩余十个空闲块 然后就会去把 801，802 到 810 这十块分配给这个文件 假如说，现在还剩下一块了 在第一组里还剩下一块了。 那么这一块呢，我们可以看到是 820 那么 820 是个空闲块，但是呢 我暂时在 820 这个空闲块里，记录了一些有用的信息 它记录了哪些有用的信息呢？我们来看一下 在第一组的第一块里头 实际上是记录了，第二组的 所有的空闲块的块号和总的空块的数量，也就是一百块 啊，再加上从 701 到800 这些数字，记录在了每一个字段里头 那么我们可以接着，啊 第二组的第一块，因为它是空闲的嘛，我们就把它用来记录 第三组的所有的空闲块的块号，当然也包括了总的啊块数 那么最后这一组，因为它是最后一组了，所以它后面没有啊 更多的空闲块了，所以最后这一组的第一块我们就没有 啊，设置。 那我们的倒数第二组，它的第一块记录了最后这一组的 啊，空闲块的块号，那么这里头大家看到是 99 块。 那么 99 块但是我们可以看到 它把最后这一组的 99 块的块号记录在了 若干字段里头，但是呢这个字段它写的是 0， 因为没有这一块，那么 0 就是我们作为判断已经是最后一组的、 这样的信息的一个标志 好，我们接着来看刚才我们说到的场景 在第一组里头还剩下

一块，那么这一块 我要如果分配给文件，因为它是空闲块 那么它原来里头保存着，第二组的块号的这些信息就会被覆盖掉 所以要在把它分出去之前，那么把 第一组的最后这一块或者第一块 它的内容拷贝到专用块里头，所以要，要有这么一个拷贝的工作 因此那么专用块的内容呢，实际上就是指向了第二组的所有块的块号 这是分配的时候。 那么如果回收的时候也是一样 当还回来的空闲块在第一组里头，放在第一组里头 如果第一组的空闲块已经超过了一百块 那么它就变成了第二组，那么多余的这些块就变成了第一组 那么于是呢，要把专用块的内容拷贝到这第一组的 啊我们说第一组的什么呀，这个第一块里头啊 然后呢，再把第一组的相关的信息 再写到专用块里头，实际上就是回收的时候要做的处理 那么这就是成组链接法。 Unix 就产用了这样一个啊 成组链接法的方法呢，来管理各种各样的空闲块 这里给出的是 成组链接法的分配算法，这里 给出的是回收算法啊，大家呢，结合刚才我的讲解呢 再看一下算法，就能明白成组链接法的一个基本的设计思想了