伙伴系统 BUDDY SYSTEM

下面我们介绍伙伴系统，伙伴系统呢 实际上是

伙伴系统

Linux低层内存管理采用

一种特殊的“分离适配”算法

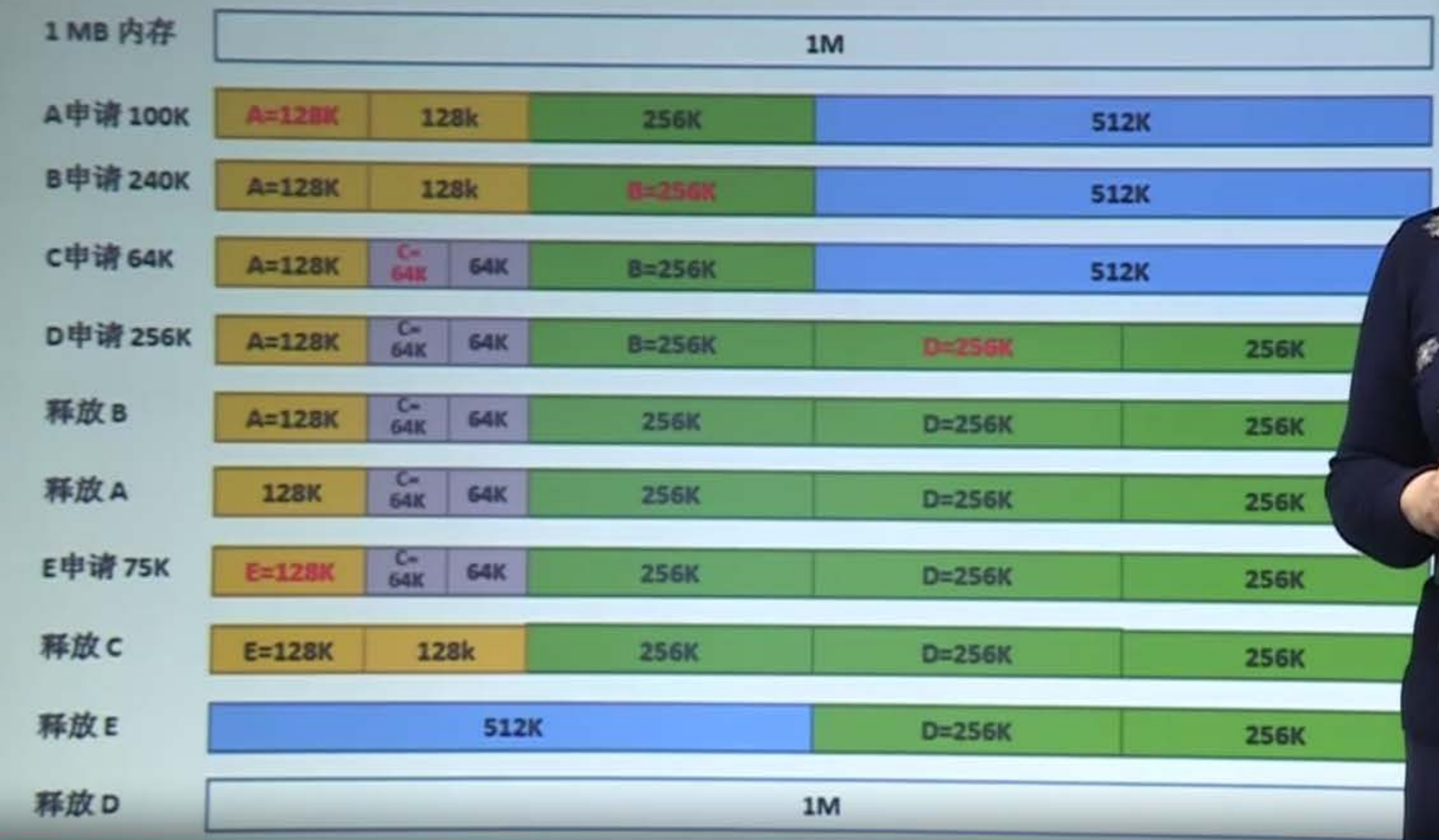
- 一种经典的内存分配方案
- 主要思想：将内存按2的幂进行划分，组成若干空闲块链表；查找该链表找到能满足进程需求的最佳匹配块
- 算法：
  - 首先将整个可用空间看作一块： 2^U
  - 假设进程申请的空间大小为 s，如果满足

    $2^{U-1} < s <= 2^U$，则分配整个块

    否则，将块划分为两个大小相等的伙伴，大小为2^{U-1}
  - 一直划分下去直到产生大于或等于 s 的最小块

s 的这样一个空闲块 那么下面我们给一个例子

伙伴系统的例子



0:00 下面我们介绍伙伴系统，伙伴系统呢 实际上是 Linux 内核底层内存管理所采用的方案 它是一种经典的内存分配算法 这种内存分配算法呢它是一种特殊的 分离适配的算法 我们简单地来看一下伙伴系统的主要设计思想 它是将内存 按 2 的整数次幂进行划分，形成了很多的空闲块 再把这些空闲块组织成链表 在进行内存分配时去查找该链表 在其中找到能够满足进程需求的一个最佳匹配块 我们简单地介绍一下它的算法的设计 首先呢，先将整个可用的空间 看成是一个大块，那么它是 2 的整数次幂 2 的 U 次方，比如说。那么假设进程 它申请的空间呢，大小是 s，那我们就来看一下 是不是满足这样一个条件。 如果 s 比 2 的 U 次方小，又比 2 的 U 次方 除以一半的空间要大的话，那么就把整个这一块 分配给这个进程，否则的话呢 就把这块空间一分为二，划分成两个大小的 相同的伙伴。 那么这就是伙伴的 这个来历。 这两个大小相同的 空闲块呢是作为一个伙伴，因为它是从一块分割出来的 好，那么继续进行相应的查找，如果 进程所需要的空间 s 又比这一半空间呢又小 又比它一半又大，就把这一半分配给它 如果不行，再继续划分，一直划分下去直到产生了一块 大于或者等于 s 的这样一个空闲块 那么下面我们给一个例子 首先我们先把整个可用空间看成是一个整块 大小为 2 的整数次幂，现在呢进程 A 要申请 100K 那 100K 呢，如果我们把 1M 空间一分为二 那么每一半是 512K，所以我们 1M 肯定要比这个 100K 要大，那么 512K 呢 也比 100K 大，所以呢我们就要把这 1M 空间一分为二 形成两个 512K 的伙伴 那么就要再看第一个空闲块，512K 512K 呢它的一半呢是 256K，所以呢 我们看 256K 又比 100K 大，所以我们要继续划分 把 256K 分成了两个 128K 那我们再来看一下，我们取第一个 128K，那我们发现 128K 的一半呢是 64K，那么 100K 比 64K 要大 又比 128K 小。 所以我们就把这 128K 分配给进程 A 那么系统中现在就剩下了 3 个空闲块，都是 2 的整数次幂 一个是 128K，一个是 256K，一个是 512K 那我们把它做成一个空间块的链表。 那么进程 B 假设要申请 240K，那我们在空间链表里中查找 找到一个能满足这个进程 B 的需求的一个最佳的匹配块，那么这就是 原来的其中的一个 256K 的，那么我们就把 256K 分配给这个进程 B 进程 C 需要一个 64K 在剩余的空闲块链表当中我们继续查找 我们有一个 128K，64K 呢 正好是 128K 的一半，所以呢我们就把 128K 给它一分为二，其中的 64K 分配给进程 C 剩下的一个 64K 的空间块，加入到这个空闲块的链表里头 进程 D 需要一个 256K 那我们继续在刚才的空闲链表中去查找 那么我们发现前面没有足够合适的最佳匹配块了，我们就把后面的这个 512K 一分为二，把 前面 500，256K 分配给进程

D，然后剩下一个 空闲的 256K。 那么这个时候如果进程 B 结束，归还了所需要的空间。 那我们来看一下 原来进程 B 在这个位置 那么归还它的空间呢，就把这 256K 还给这系统，所以系统 现在有这样几个空闲块，这有 64K，这有一个 256K 这有一个 256K，但是这两个 256K 呢大小是一样 但是它们俩是不同的这个，这个空闲块，它们俩也不是个伙伴关系 那么进程 A 结束，还回了 128K 这样的话又多了一个空闲块，这个时候进程 E 来了，申请了一个 75K 那么 75K 的话呢，我们在刚才的空闲块表 链表里就去查找，就把前面这 128K 又分配给了进程 E 进程 C 结束之后呢 这个时候进程 C 占的这 64K 还回给系统。那么归还到 空闲块链表的时候，我们就会看到这个这 64 K 它的伙伴，另外一个 64K 也是空闲的，所以两个伙伴就 合并，变成了一个更大的空闲块。 那么这个时候我们可以看到 两个 64K 合并起来变成了一个 128K 的空闲块 那么进程 E 结束 进程 E 结束的时候呢还回了 128K 和刚才的 128K 正好又构成了一个伙伴，所以它们两个 空闲的伙伴块呢，就合并成一个更大的空闲块 然后我们又继续发现，还有一个伙伴 256K 和它合并以后的 256K 构成的是一个伙伴关系，所以这两个伙伴又合并成一个更大的空闲块，所以最后我们得到了一个 512K 的空闲块。 所以到现在为止呢，系统 中呢我们可以看到，有一个进程 D 占用了 256K 然后有一个很大的空闲块，是 512K 和一个 256K 的空闲块 那么如果进程 D 结束了，那么我们可以看到进程 D 所占的 256 K 和它的伙伴变成了一个更大的空闲块 512K，然后又和它的伙伴又 合并成一个更大的空闲块，就最后就得到了一个 1M 的一个很大的可用空间 那么伙伴系统呢，它实际上它的主要的思路是 每次分配的时候，我们不要把一个大的空闲块拿来给它分割 拿一个比较合适的空闲块 而这样做的话呢，就使得系统中会保持很多大的空闲块，那么当然了，我们也可以看到如果 分割完之后，剩余的部分，以后呢和这个前面分割的部分呢形成一个伙伴关系。 当 原来分出去的还回来之后，就可以和刚才剩余的部分又合并成一个更大的空闲块 这样的话系统中可以始终保持一些大的 空闲块，这就是伙伴系统的一个主要的设计思路