# Linux内存分配之伙伴系统

## 背景

内存碎片是内存分配中产生的不可忽视的问题，内存碎片的概念很复杂，我目前只理解一点，就是当我在malloc申请内存的时候，如果找不到符合我要求的连续的内存空间，那么malloc将会失败。如果不理会内存碎片，我们的系统将会很快不可用，因为无法申请稍微大一点的空间了。

## 2.目的

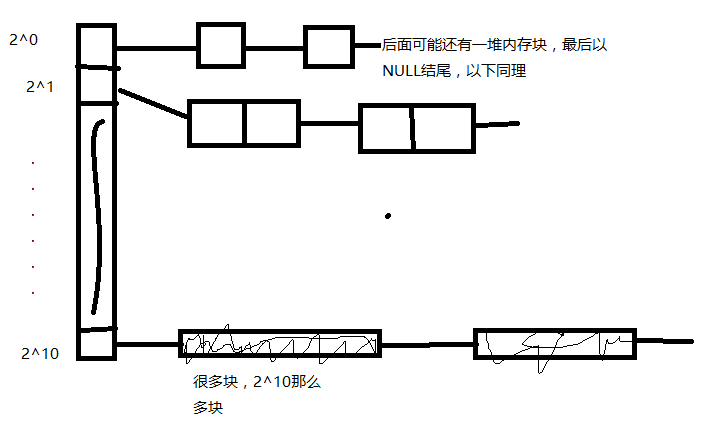
为了解决内存碎片的问题，使内存保持更多连续的空间，不要四处都是小空间。伙伴系统就应运而生了。

## 伙伴算法

伙伴算法能使内存拥有更多的连续空间。伙伴系统的宗旨就是用最小的内存块来满足内核的对于内存的请求。

### 3.1基本概念

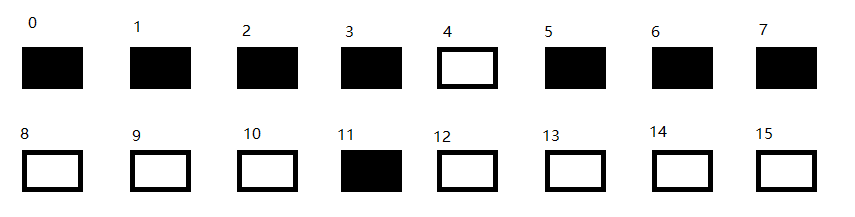
伙伴算法，就是把内存分为2^order种大小类型，order一般为10（看Linux内核代码的宏定义就知道了，这次时间仓促，就不结合代码说了，只讲概念）。也就是有11种大小类型的内存块，2^0/2^1......2^10，单位是页（一页为4K），他们都挂在链表上。示例图如下：



那么所有的内存块都分别挂在了不同大小的order上。

看到这里，大家肯定会想，这特么跟伙伴两个字有毛的关系？是的，看到这里并没有什么关系，如果相同单位（2^N）的相邻的两块内存一块分配了内存，一块没有分配，那么他们就是伙伴关系。举个栗子：

假如我的内存只有16页，如下图：



黑色代表分配了的内存，白色代表空闲内存。那么伙伴的含义由上图可以解释了。

1. 对于2^0大小的内存块， 0/1不是伙伴，因为两个都分配了内存。依此类推，4/5是伙伴，10/11是伙伴。那么他的伙伴关系我可以表达为：

Order0：4[4.5] ,10[10/11]

2. 对于2^1大小的内存块，4-5/6-7是伙伴，8-9/10-11是伙伴，可以表达为：

Order1：4[4-5,6-7] ,8[8-9,10-11]

对于2^2大小的内存块，表达式为

Order2: 0[0-1-2-3/4-5-6-7], 8[8-9-10-11/12-13-14-15]

3. 对于2^3大小的内存块，没有伙伴。

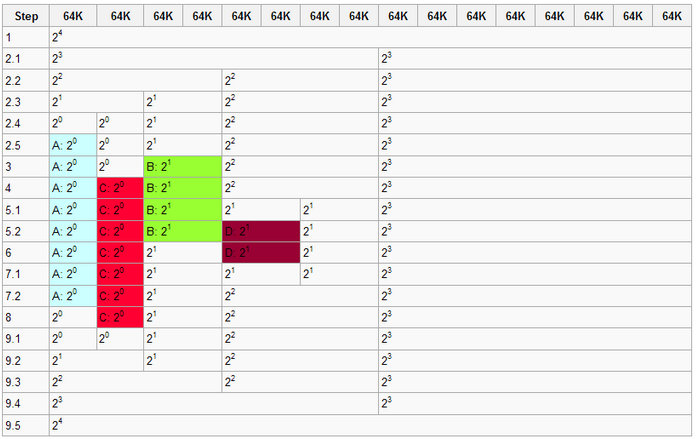
4. 对于2^4大小的内存块，这里的例子一共就这么大，自然没有伙伴。

那么确立伙伴关系，到底怎么发挥作用呢？简单地来说，如果一个内存被释放了，那么它就会检查他是否有伙伴，如果有伙伴，小的内存就会合并为大的内存。合并之后大的内存继续检查是否有伙伴，有的话继续合并为更大的内存，直到不能合并为止。

### 3.2 实际运用

最后，我们来举一个分配内存和释放内存的实际例子，通过实例一步步帮助大家理解伙伴算法。懒得写了，举例子就抄一个网上的：http://blog.csdn.net/vanbreaker/article/details/7605367

假设在最初，只有一个块，也就是整个内存，假如为1M大小，而允许的最小块为64K，那么当我们申请一块200K大小的内存时，就要先将1M的块分裂成两等分，各为512K，这两分之间的关系就称为伙伴，然后再将第一个512K的内存块分裂成两等分，各位256K，将第一个256K的内存块分配给内存，这样就是一个分配的过程。下面我们结合示意图来了解伙伴系统分配和回收内存块的过程。



1 初始化时，系统拥有1M的连续内存，允许的最小的内存块为64K，图中白色的部分为空闲的内存块，着色的代表分配出去了得内存块。

2 程序A申请一块大小为34K的内存，对应的order为0，即2^0=1个最小内存块

   2.1 系统中不存在order 0(64K)的内存块，因此order 4(1M)的内存块分裂成两个order 3的内存块(512K)

   2.2 仍然没有order 0的内存块，因此order 3的内存块分裂成两个order 2的内存块(256K)

   2.3 仍然没有order 0的内存块，因此order 2的内存块分裂成两个order 1的内存块(128K)

   2.4 仍然没有order 0的内存块，因此order 1的内存块分裂成两个order 0的内存块(64K)

   2.5 找到了order 0的内存块，将其中的一个分配给程序A，现在伙伴系统的内存为一个order 0的内存块，一个order1的内存块，一个order 2的内存块以及一个order 3的内存块

3 程序B申请一块大小为66K的内存，对应的order为1，即2^1=2个最小内存块，由于系统中正好存在一个order 1的内存块，所以直接用来分配

4 程序C申请一块大小为35K的内存，对应的order为0，同样由于系统中正好存在一个order 0的内存块，直接用来分配

5 程序D申请一块大小为67K的内存，对应的order为1

5.1 系统中不存在order 1的内存块，于是将order 2的内存块分裂成两块order 1的内存块

  5.2 找到order 1的内存块，进行分配

6 程序B释放了它申请的内存，即一个order 1的内存块

7 程序D释放了它申请的内存

   7.1 一个order 1的内存块回收到内存当中

   7.2由于该内存块的伙伴也是空闲的，因此两个order 1的内存块合并成一个order 2的内存块

8 程序A释放了它申请的内存，即一个order 0的内存块

9 程序C释放了它申请的内存

   9.1 一个order 0的内存块被释放

   9.2 两个order 0伙伴块都是空闲的，进行合并，生成一个order 1的内存块m

   9.3 两个order 1伙伴块都是空闲的，进行合并，生成一个order 2的内存块

   9.4 两个order 2伙伴块都是空闲的，进行合并，生成一个order 3的内存块

   9.5 两个order 3伙伴块都是空闲的，进行合并，生成一个order 4的内存块

总之一句话，分配内存之后，会产生一个伙伴。释放后内存后，会查找是否有伙伴，有就合并，合并之后还有就继续合并，这样内存块就容易维持得比较大了，而不是很多很小的内存块。

欢迎大家提出疑问。