VOS内存分配算法\_buddy算法实现

作者：[156439848@qq.com](mailto:156439848@qq.com); vincent\_cws2008@gmail.com

时间： 2020-09-07



如上图，基于buddy分配算法实现的动态内存分配模块，buddy算法的核心是减少碎片的产生，每次把空闲内存划分成多组固定大小的内存池链表，大小是按幂次方递增，例如：2^0链表存储的数据都是1个页，每页的大小通常为1K, 2K, 4K等等。这样释放的页基本都是次方对齐，没任何碎片，因为碎片都被直接浪费在申请块中，如果realloc申请，剩余的空间又足够，这样就不浪费。

如上所述，buddy算法最大的缺点就是浪费内存，例如申请1个字节，必须1个页，如果申请2个页+1字节内存，就直接分配2^2=4页内存来存储。

这里我们介绍了buddy算法的优缺点后，我们继续介绍实现算法：

设计如何实现这个buddy算法前，思考了很多，包括各种方法，

例如：

（1）直接在内存块里格式化一个内存空闲控制块头部和内存已分配控制块头部，这样实现最简单，不过有个问题，就是如果上下块写越界，很容易破坏链表，导致整个破坏。

（2）也想过使用位图实现，不过直接查找位图或者算法查找位图，想不到更快的查找位图方法。

（3）最后基于第一个想法，不过不把页控制块放到页中，而且使用控制块和页块平行数组的方法来互相映射（偏移位置一致），这样就算页写穿了也不会破坏到控制块的信息，而且这个浪费的内存也不算多，例如每页1K字节，控制块信息（双向链表+块控制信息）总共才4个32位整数，16字节，估算花销也大概1.5%, 如果每页是4K字节，花销0.39%。平行数据有个好处，就是释放时可以不用遍历链表。

1 堆的创建：

堆创建就是格式化用户提供的内存空间，格式化数据如上图，包含堆头，控制块数组和页数组，因为刚创建时所有内存都是空闲的，就必须把空闲的划分到对应的内存池链表中。

2 申请内存实现：

申请内存就是在指定的堆里找指定大小空间的内存块，如果对应的内存池链表中找不到任何块，就往上一层申请，重复直到上层有或者顶层为空才退出，如果上层有对应内存池，就必须像二叉树那样一分为二，裂开到本层后，就可以申请成功，否则申请失败，返回NULL.

因为多组内存池空闲页链表，申请成功时就直接从某个组的链表头扯出第一个空闲页就不可以，不使用遍历组内的链表元素，不过裂开的过程就是垂直搜索过程，不过内存组个数最大就是10个，最坏的全部搜索也是10个链表操作，也非常快。

3 释放内存实现：

释放其实也要考虑很多东西，包含释放内存指针指向的地址范围是否合法，如果知道释放内存块占用多少页，这个就得称赞平行数组的好处，根据当前页的偏移个数找对应控制块的信息，对应控制块记录了已经分配的大小和页数，还有分配状态，这样就可以直接释放，不用遍历对应的链表，速度也相当快。

释放的2^n个页和相邻块2^n页是空闲的，还要判断是否可以合成2^(n+1)个页，注意：不是相邻的都能合并成上层的2^(n+1)的页，还得看是否是2^(n+1)地址对齐，如果某个2^n个页的相邻页恰好是上层1页的右半部和上层2页的左半步，也不能合并。

4 测试用例说明：

主要测试数据结构里的元素是否合法，同时检测总和数据是否一致。