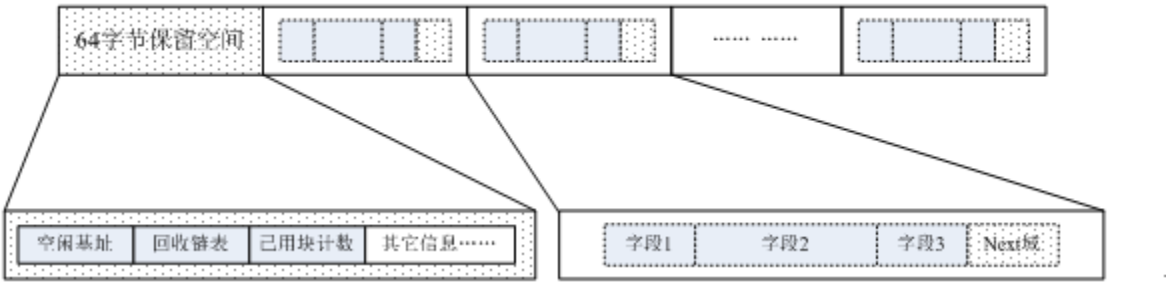
自管理内存分配的哈希表

1.分配释放管理



在链表结构中，当需要分配新块时，需要按如下步聚进行：

a) 回收链表指针值是否为0？

i. 如果不是，则分配回收链表指针指向的结点NodeA，同时设置回收链表指针为NodeA.Next，分配完成，跳到c；

ii. 如果是，则跳到b。

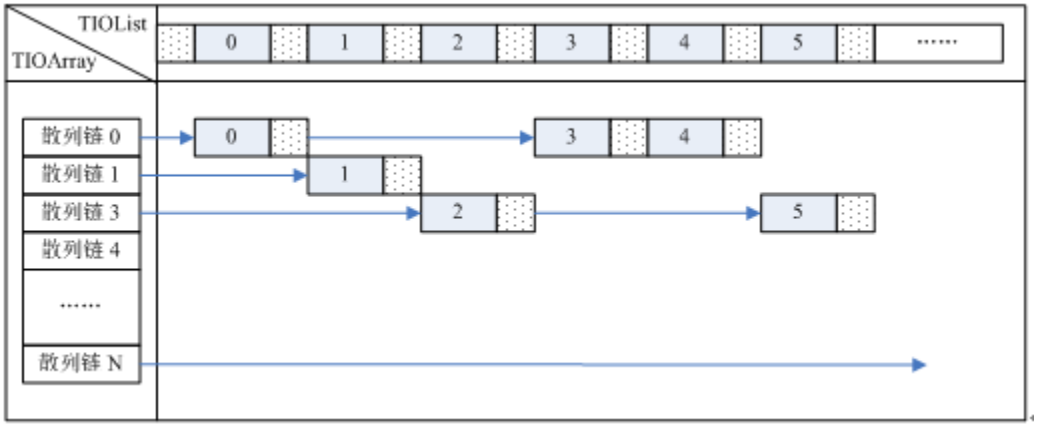
b) 空闲基址指针值是否小于等于文件大小-数据块的大小？

i. 如果不是，则分配失败，退出；

ii. 如果是，则分配空闲基址指针指向的结点NodeB，同时设置空闲基址指针为NodeB+数据块的大小。这里需要注意的是，因空闲基址指针初使值为0，第一次分配时，应分配地址值为64的块。

c) 如果分配成功，则对已使用块计数器加1。

2. key-value映射管理




```
/************************************************
+------------------------+-------------+
| struct TableHead       | 64 bytes |
+------------------------+-------------+
| struct Array           | m_TableLen |
+------------------------+-------------+
| struct Entry           | nodeTotal |
+------------------------+-------------+
************************************************/
```

hashtable分为array内存块和list内存块，以及管理块TableHead

TableHead负责rehash，hash swap，hash algorithm (MurmurHash3)，空闲节点比例，空闲链表，回收链表等操作

array内存块负责hash桶数组。

list内存块负责双向链表维护。

hashtable.txt