Malloc Lab

王振宇

Lab解读

- 在本次实验中需要实现一个动态内存申请器，在mm.c中编写如下函数和辅助其实现的静态函数。
- `mm_init malloc free realloc calloc mm_checkheap`

- Score = 0.6*Util_score + 0.4*Thru_score + Consistency + Style
- Util_score有分/满分需要达到平均70%/90%的内存利用率
- Thru_score有分/满分需要达到4000/14000Kops,本地和autolab有偏差
- Score低于50将变为0，从mm_textbook.c出发，平衡利用率和吞吐量！

（但实践上应该会是在不断地优化利用率，吞吐量并不是评分的关键限制。）

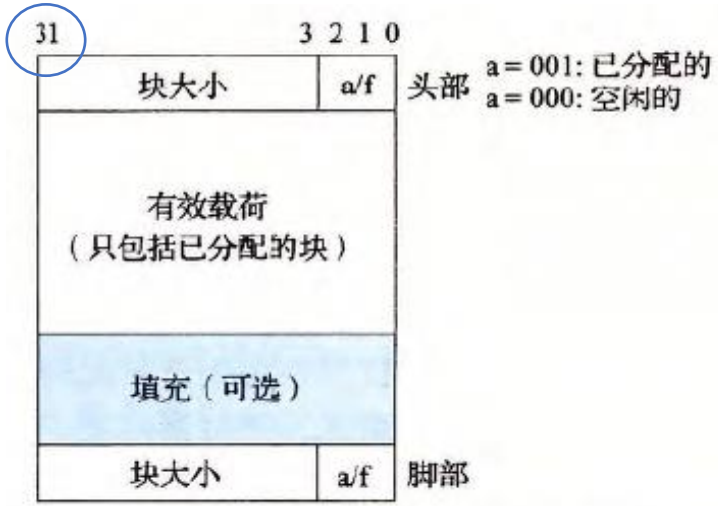
NOTE

- 在**mm_init**中需要重新初始化所有全局指针。
- 在**malloc**中返回指向负载的八字节对齐的指针。
- **calloc**和**mm_checkheap**不参与mdriver的评分（但还是要写的），前者一个简单正确的实现就好，后者推荐先写好，有助于调试代码。
- 运行**mdriver**进行评分时记得注释掉#define DEBUG，否则对thru测试有影响。

一些可供参考的思路——块的结构

mm-textbook已实现的部分:带脚部的隐式空闲链表(32位头部脚部)

块是以下图的形式组织的，free时通过检查脚部和头部判断上下块的空闲情况。



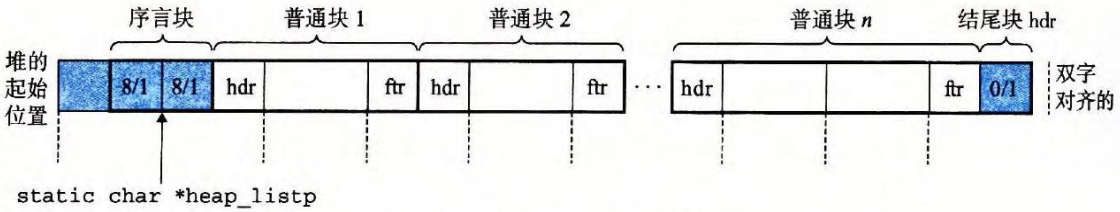
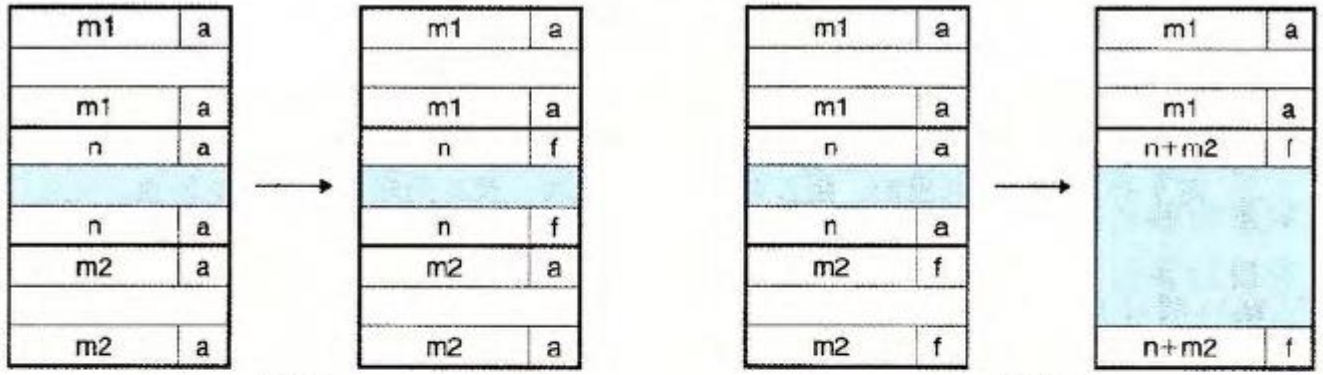
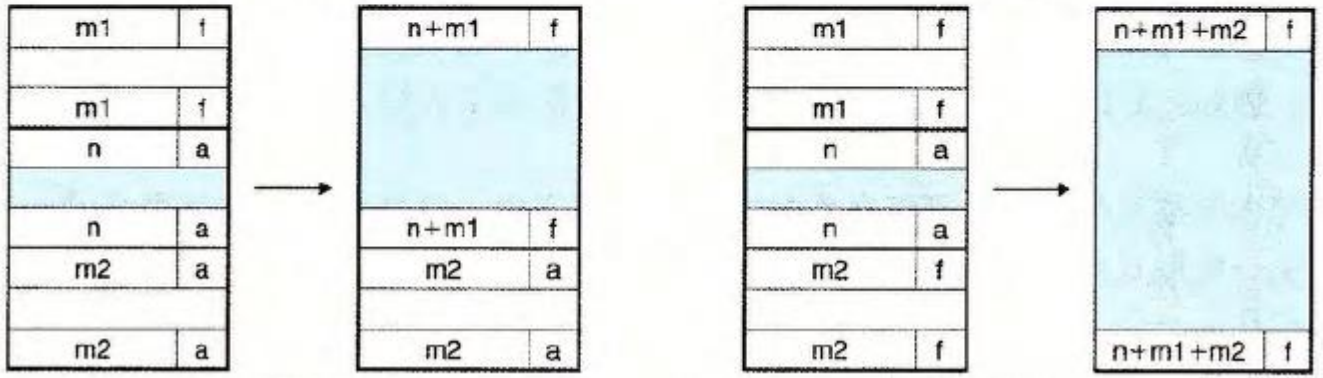


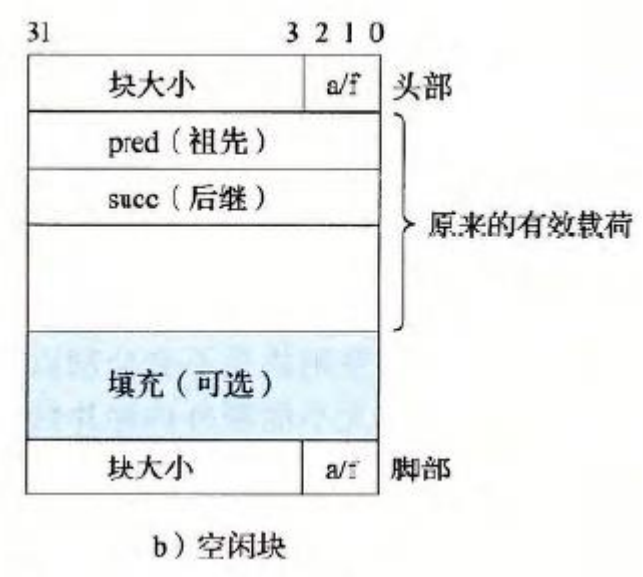
图 9-42　隐式空闲链表的恒定形式





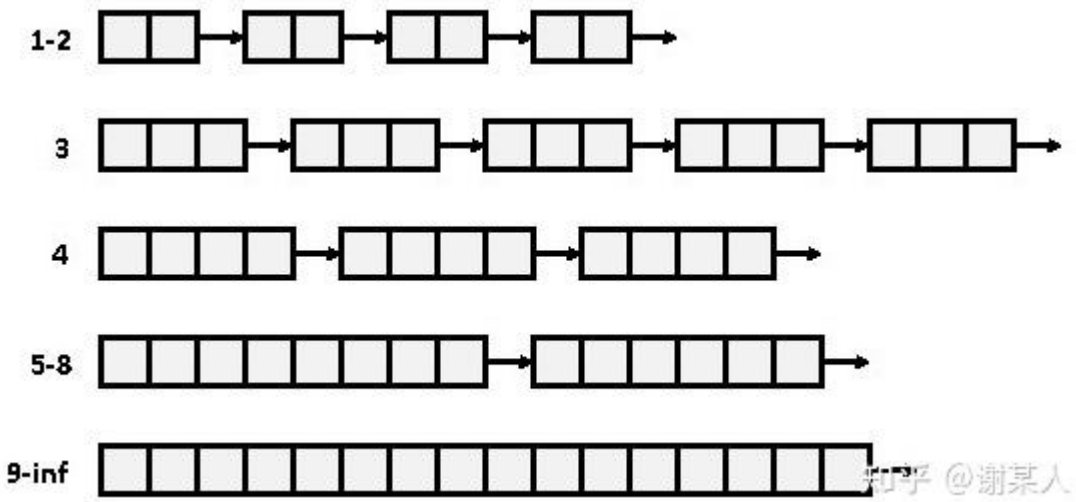
一些可供参考的思路——块的结构

- 1.1. 加速空闲块查找，修改为显示空闲链表
- 1.2.去除分配块的脚部，利用a/f的标记位记录上一块的空闲状态完成合并空闲块过程（空闲块在free时补充脚部）
- //去脚部对padding大小平均意义下无影响，省去了分配块时的脚部空间,可以较大地提高util



一些可供参考的思路——空闲块的组织

- 2.1采用分离适配链表管理空闲块，极大地提高thru.对分离适配链表使用firstfit的结果接近于使用bestfit,也能提高util.
- 2.2压缩链表指针为偏移,节省最小块的空间.
- 2.3改为分离适配链表带来了另一个好处，可以采取适应性的fit策略，小空闲块的数量较多，但空间不大，可以采用firstfit,大空闲块数量较少，但占用空间很大，不优的fit策略可能会较严重地消耗空间，可以采用bestfit，用thru换一些util.//也可以维护一个部分有序的链表兼顾util和thru。



一些可供参考的思路——进一步的优化

- **我最后的tricks**
  - 3.1 对最多的最小分配块再次进行优化(MiniBlock)
  - 3.2 (极度)简化的Slab机制

- **可能可行或优化较小的方法**
  - 4.1 改进Fit和Place策略

    best_fit并不总如其名一样best，best_fit实现的功能仅仅是找到空闲块中大小和需求差距最小的块，但这样也很容易产生一些小碎片，有没有更优秀的fit方法？

    Place总是将分配块放在空闲块的头部或尾部，根据块大小采取不同的策略？(在bestfit下就没有办法做这样的事了).
  - 4.2采用更高效的数据结构(红黑树等平衡树)来组织空闲块

    但用处不大(因为瓶颈在于util而非thru),要求写出一个复杂的数据结构也非lab本意。//怎么只使用很少的空间实现这些数据结构?

- **Evil Trick**
  - 过程中我们引入了一些参数！CHUNKSIZE?分离适配链表的分列形式? firstfit和bestfit的分界点？Slab的大小？调参(炼丹)

Debug

- 想清楚实现哪些,如何实现,再开始写代码!
- 逐个加上优化,一次把所有优化加上很难跑通!
- 多封装正确的宏，定义宏时确认正确性!
- 小心指针加减与转换!
- 调试命令在7 The Trace-driven Driver Program 和11 Hints里写的很详细