malloclab实验报告

整体方案

采用显式分离空闲链表+分离适配方法来完成这个实验。

具体地，该方案会维护一个空闲链表的数组，每个数组元素是一个大小类中的空闲链表——它串起来了一系列空闲块，能高效地实现内存的分配与释放。每个分配块由头部、填充内容和脚部构成，而每个空闲块由头部、指向前一个空闲块的指针、指向后一个空闲块的指针、空闲内容和脚部构成。

辅助函数

insert

1. 选择到所需的链表。
2. 一直向前遍历，直到遍历到大小比他大的块或者遍历完链表就结束。
3. 根据前后块是否是空指针来设置关系。

delete

1. 选择到所需的链表。
2. 如果空闲块的前指针或后指针指向的地址已经跨出当前堆边界，则直接将其设置为空指针。
3. 否则判断空闲块的前指针和后指针状态再做处理。

extend

1. 设置头部、脚部以及新的结尾块。
2. 插入新的空闲块并检测是否可以合并某些区间。

coalesce

1. 检查前后块的分配状态，分类处理并将合并后新的块插入链表。
2. 两者都已经分配，则不合并。
3. 前面已分配，后面空闲则合并后一块。
4. 后面已分配，前面空闲则合并前一块。
5. 都是空闲的，则全部合并。

主函数

mm_init

1. 创建MAXLIST个list，设置堆起始位置。
2. 初始化空闲列表，创建空堆并放入序言块，失败返回-1。
3. 最后给空堆拓展INIT字节，失败返回-1。

mm_malloc

1. 空指针不用释放，直接返回。
2. 将头部脚部设成未分配状态。
3. 将空闲块加入链表并判断是否可以合并。

mm_free

1. 大小为0直接返回。
2. 调整块的大小，至少为16字节。
3. 遍历分离空闲链表以获得合适的空闲块，如果没有找到一个合适匹配块就拓展堆。
4. 根据调整后的大小放置这个分配块。

mm_realloc

1. 如果指针指向的内存块是空的，就相当于直接分配空间。
2. 如果大小等于零就相当于直接释放。
3. 首先对齐大小，接着进行判断：如果原本的空间比现在要设置的空间大，那么就直接返回原指针。
4. 如果下一块没有分配或者大小为零，就把当前块与下一个空闲块合并，必要时可以扩展堆。如果所有方法都不满足，则将之前的内容复制到当前块。

一点思考

刚开始就没有想着写各种树，因为看了下mm-tree.c觉得太复杂了，而且示例代码只支持32位机器运行，无法进行参考。后来看到课本上讲GNU使用的malloc包采用了分离适配方法，于是最终设计了将显式分离空闲链表与分离适配方法相结合的方案，最后的结果也的确如课本所述一样较为高效。