OSlabs Report

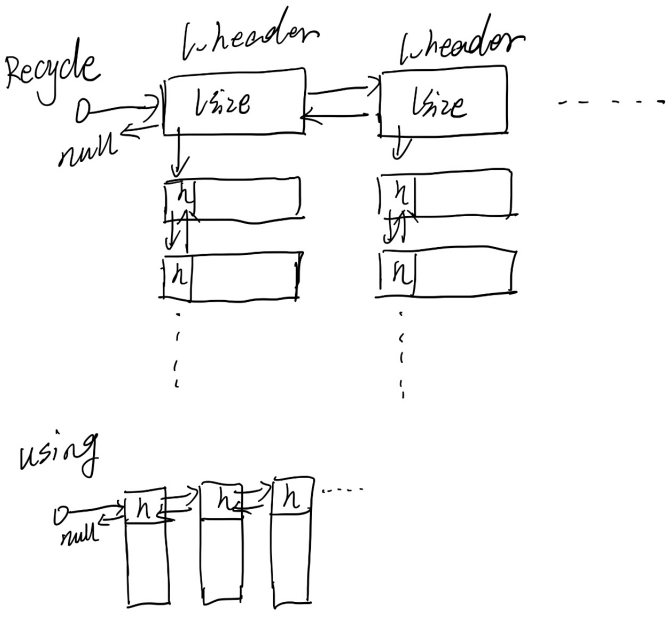
姓名：张灵毓　　**学号：**171240524

2019 年 4 月 2 日

1　L1:kalloc/kfree

1.1　框架设计

框架设计没有区分小内存分配回收和大内存分配回收。
分配空间都分配 2^kB
采用 slab 的思想, 不过对空闲的空间不进行合并, 用两个 list:using 和 recycle 分别表示当前在使用的内存之和之前使用过并被回收的内存, 如图所示,recycle 所指的链表的每一个 l_header 中记录其所指的内存链表中每一个内存块的大小 lsize, 而 using 指针则直接指向若干个正在使用的内存块



分配: 先根据所需要的内存大小选取适当的 k 满足 $size > 2^{k-1}$ 且 $size \leq 2^k$ 记 $2^k = alloc\_size$, 然后遍历 recycle 链表, 找到满足条件的 lsize = alloc_size, 且 $l\_header- > begin_h \neq$ NULL 的结点, 即表示此处存在未分配的内存块, 将第一个符合条件的内存块分配即可并将这一个内存块从 recycle 中删除, 插入 using 链表, 如果没有符合条件的 l_header, 那么就通过 create 同 _heap 中分配一块新的空间来使用 (空间大小为 $4 *$

$alloc\_size$, 每次分配 4 块, 提高分配效率)
内存回收: 只需要将内存块从 using 中删除, 插入到 $lsize == alloc\_size$ 的对应 l_header 指向的链表中, 留下次使用即可.

1.2 something about bug

在使用自己实现的 test 函数时, 创建一个 void * 的数组用来得到 kalloc 的返回值, 如果将这个数组定义在 test 函数外面, 即全局变量, 则无论多大, 测试能够通过 (测试了很多次...) 如果放在函数里面, 则超过一定大小就会出现神奇的问题, 不超过这个大小也可以通过测试 (测试了很多次...), 一开始也不知道为什么通过检测否决了爆栈的可能性, 从而转向考虑是否是结构体对齐问题, 检查代码中的结构体, 在 heap 中都按照 16 字节对齐, 所以没有问题, 在最近的测试中, 发现, 如果在 pmm->init 里面打印 pm_start 的值, 为正常的 0x200000, 而在 kalloc 里面打印变成了 0, 说明还是一开始想到的问题, 即爆栈了, 将静态数据都给覆盖掉了, 从而导致问题.