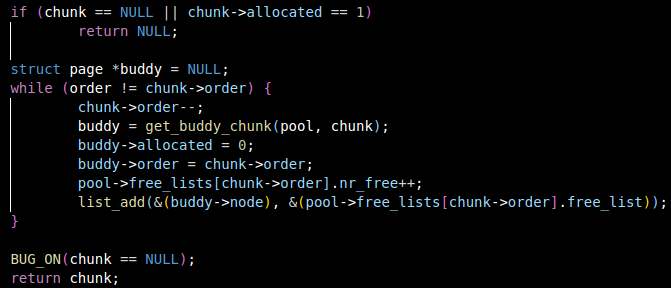
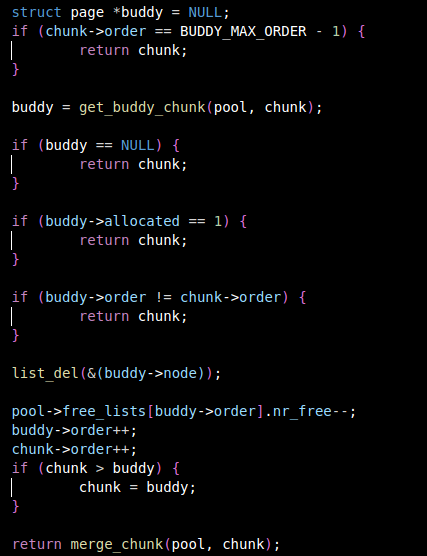
练习题1：

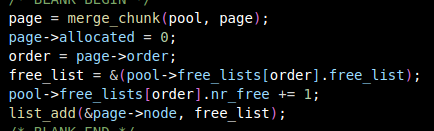
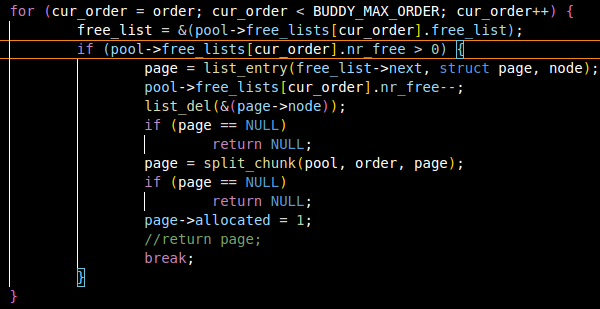
merge\_chunk： split\_chunk:



合并参考教程，采取递归方式，先判断能否进行递归调用，如果不能则直接返回该chunk，删除list中的buddy并更新相关信息，在小块合并后，如果大块能然能合并就会再次合并

分裂使用循环的方式，如果不是需要的order，就一直向下分裂，直到获取所需大小的chunk，期间需要加入对应order的free\_list

buddy\_get\_pages buddy\_free\_pages

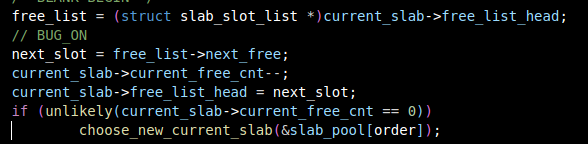
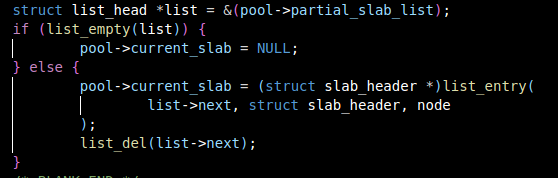


获取页从低到高遍历order，如果当前order存在free的部分，就删除结点，进行拆分, 没有则继续向后寻找。

释放页则直接进行合并，然后放回free\_list

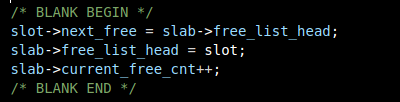
练习题2：

choose\_new\_current\_slab alloc\_in\_slab\_impl

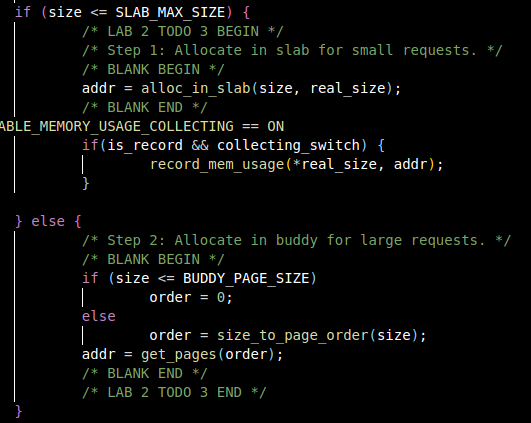


free\_in\_slab

1. Choose… 将pool->current\_slab指向空闲slab\_list中的第一个slab，然后删除掉该slab，表示已经被使用
2. Alloc… 获取当前slab中头部的空闲slot(free\_list\_head)，更新相关信息，如果当前slab空闲slot为0，则获取新的
3. Free… 将该slot的next指向原头部free\_list\_head，slab的free\_list\_head指向该slot，最后增加free的数量

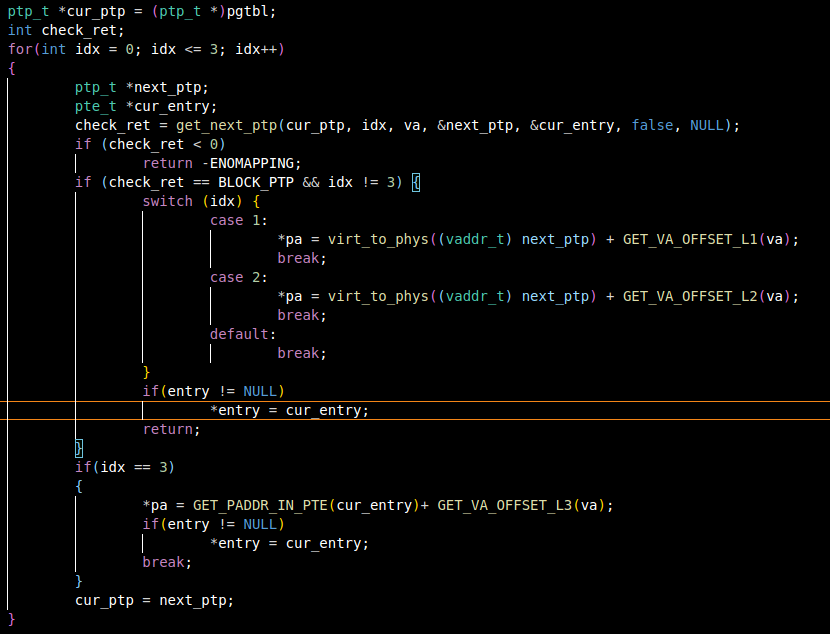


练习题3：



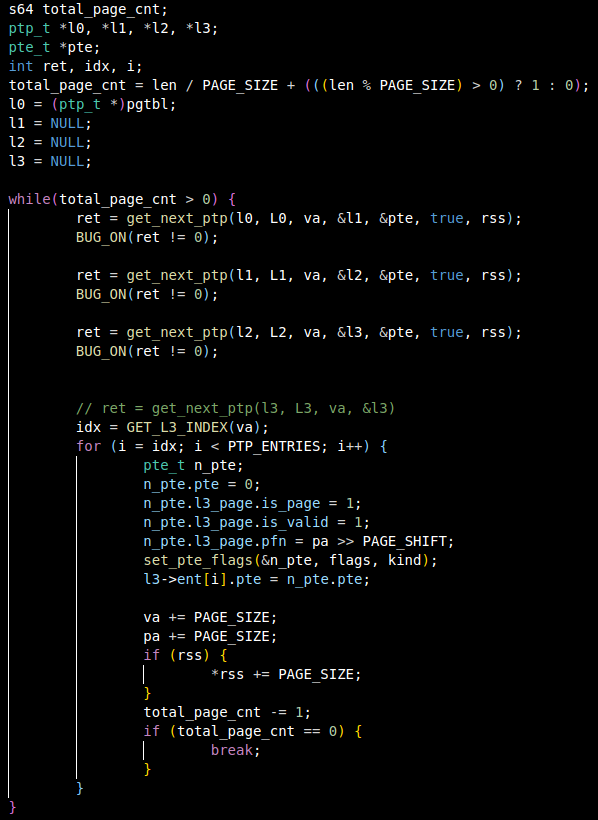
如果小于slabsize，就从slab分配内存，否则根据内存大小判断order，在buddy中分配内存

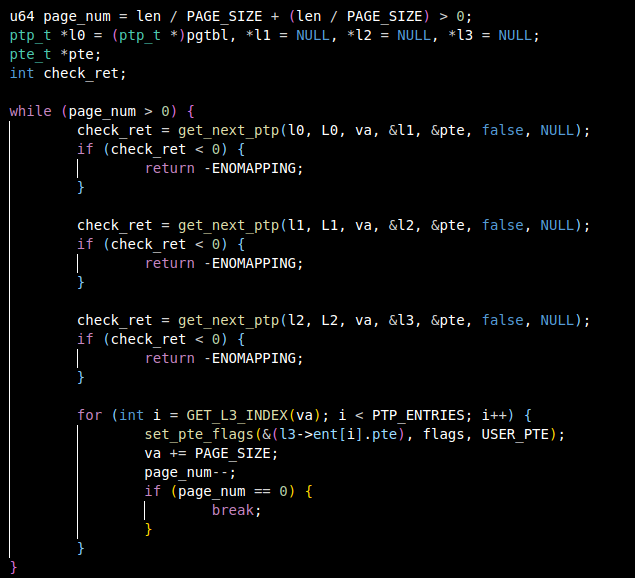
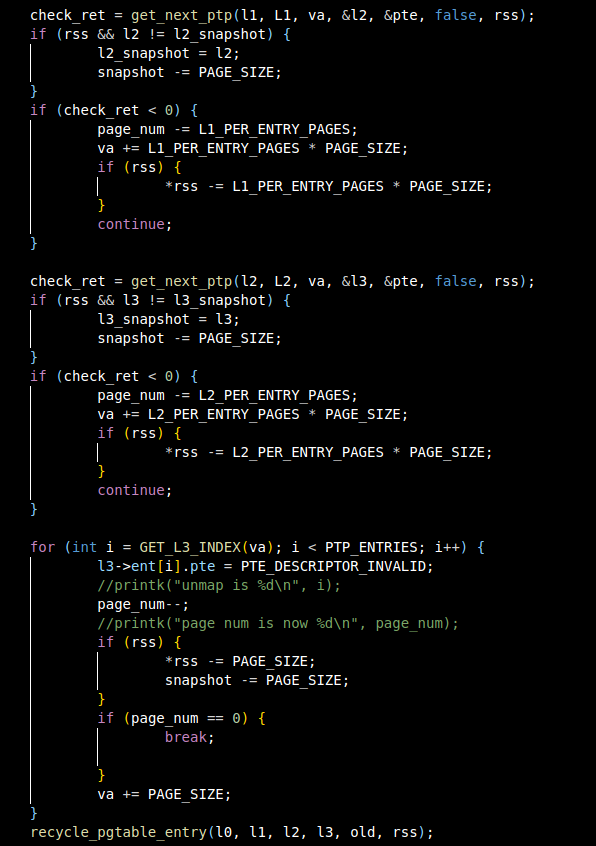
练习题4：



遍历三级页表，在第三级页表处返回对应的物理地址以及pte entry，如果1，2级页表是大页，也返回对应的数据

Map\_range\_in\_pgtbl\_common 遍历页表，在最后一级设置相关属性，添加至页表项，完成4k粒度的映射





<-部分代码片段

Unmap， 遍历页表，在第三级页表的对应位置将pte归零

Mprotect：遍历页表取到第三级pte之后，对权限进行修改

思考题5：

需要设置访问权限位：AP字段，将权限设置为只读，当出现写操作时，发生page fault

如何处理：CPU将控制流传递给缺页异常处理函数，OS发现对只读内存进行了写操作，对应区域为COW，OS会将发生异常的物理页进行拷贝，把页权限设置为读写，然后更新页表项，此时进程可以正常访问该页进行读写

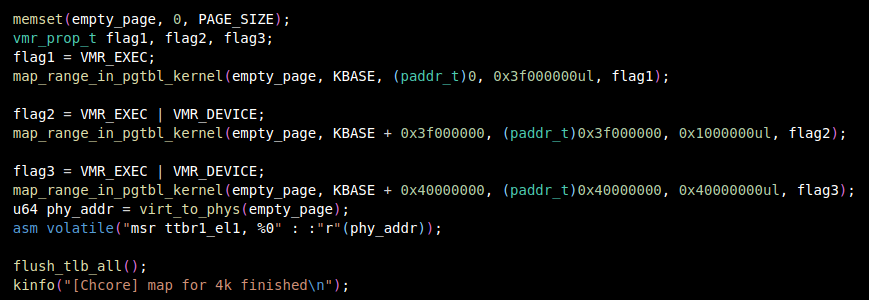
思考题6：

内存浪费：进程可能不会完整地使用大页，而未被使用的大页内存也不能分配给其它进程，导致了内存的浪费

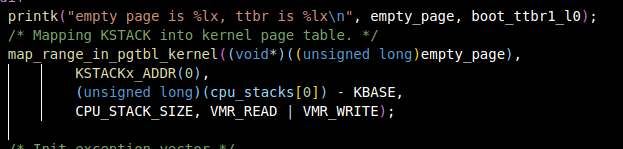
权限控制：对大页pte设置完权限之后，该块内存的权限都是相同的，无法进行细粒度的权限管理

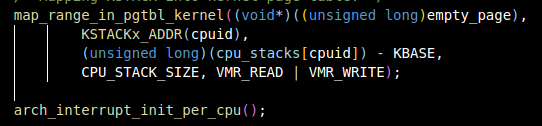
内存碎片：当大页分配过多时，可能导致页之间产生空隙虽然总量足够，但是单一空隙的大小并不足以进行分配，造成内存碎片

挑战题7：



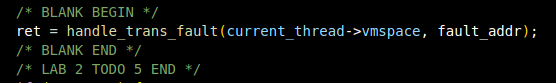
观察到main.c中全局设置了一个空白页，使用该页作为L0对kernel内存进行映射，映射完毕后通过msr指令切换L0页表





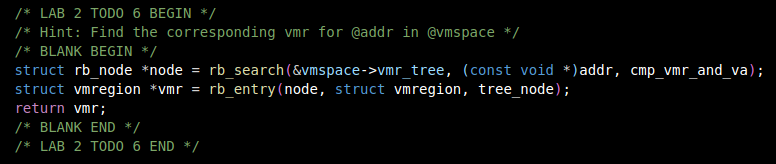
在对kernel stack进行映射的时候，将L0替换为empty\_page(需要注意boot ttbr1为低地址，需要加上KBASE，而empty page为高地址)

练习题8：



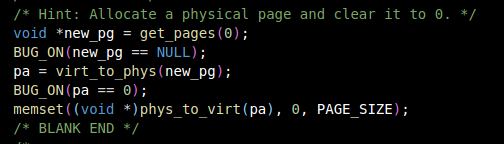
使用handle\_trans\_fault对异常进行处理

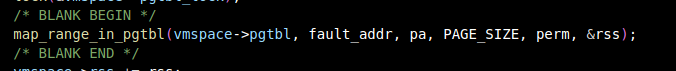
练习题9：

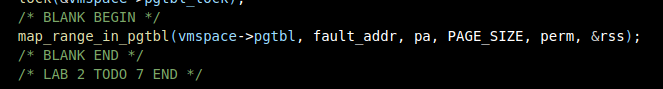


使用rb\_search函数寻找对应地址的node，再通过rb\_entry取出结构体的起始地址

练习题10：



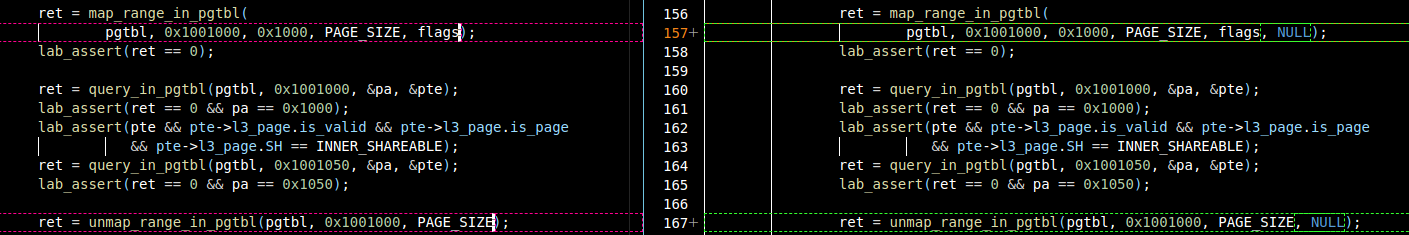




先分配对应的页，并将其置零

然后对缺少的页进行映射

挑战题11： 此处修改过lab2.c中的接口，否则基础测试的参数不正确(如下图)



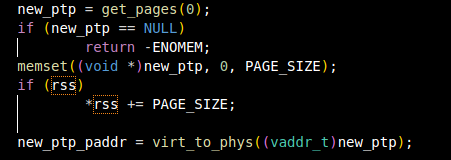
修改get\_next\_ptp map unmap对应参数，增加rss参数

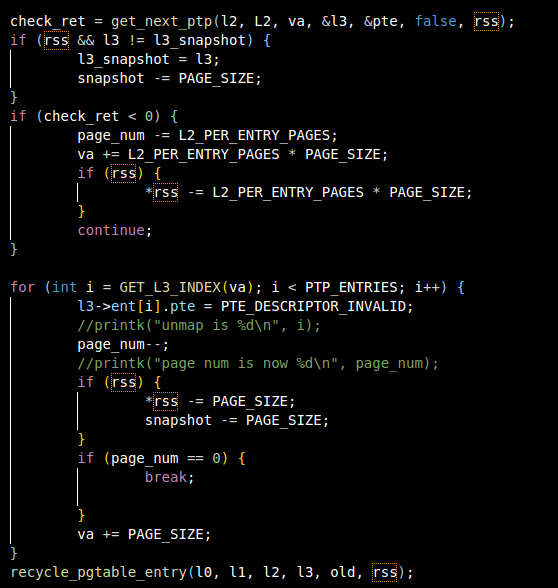
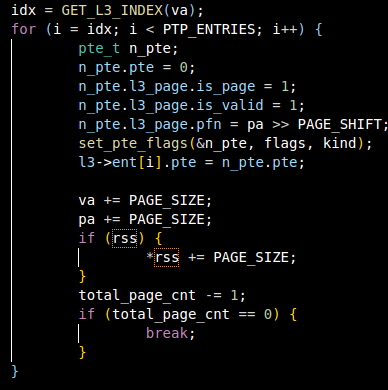
get\_next\_ptp中，如果新分配页表，相应rss += PAGE\_SIZE

map函数中，映射多少页，就对rss增加响应的大小

unmap函数中, 根据取消映射的页数，减少对应的rss，同时调用recycle\_pgtable\_entry进行页表页的回收，并添加判断，当map时分配页表但unmap时不释放页表时，会减去页表部分的rss，同理当map不分配页表，但unmap释放页表时，也会加上页表部分的rss

(下为部分片段)





Lab本地评测结果：

