CS3601 Lab2 实验报告

王梓萌 521030910015

1 练习题1

`split_chunk`、`merge_chunk` 是伙伴系统机制的重点所在，`buddy_get_pages` 和 `buddy_free_pages` 则利用这两个基本的函数完成了分配物理页并更新伙伴系统的工作。题目关键在于实现递归合并和拆分时的逻辑。

详细实现请见源代码

2 练习题2

`choose_new_current_slab`、`alloc_in_slab_impl` 分别负责为一个分配请求找到合适的slab池和空闲的slot，并维护slab分配器。`free_in_slab` 则负责在释放请求到来时进行释放操作。题目的关键在于对slab分配器中各指针的运用和维护。

详细实现请见源代码

3 练习题3

`kmalloc` 会根据一个分配内存请求的大小，调用我们刚刚实现的slab分配器或者伙伴系统，最后返回分配的物理内存的起始地址（均使用的是内核页表下的虚拟地址）。

详细实现请见源代码

4 练习题4

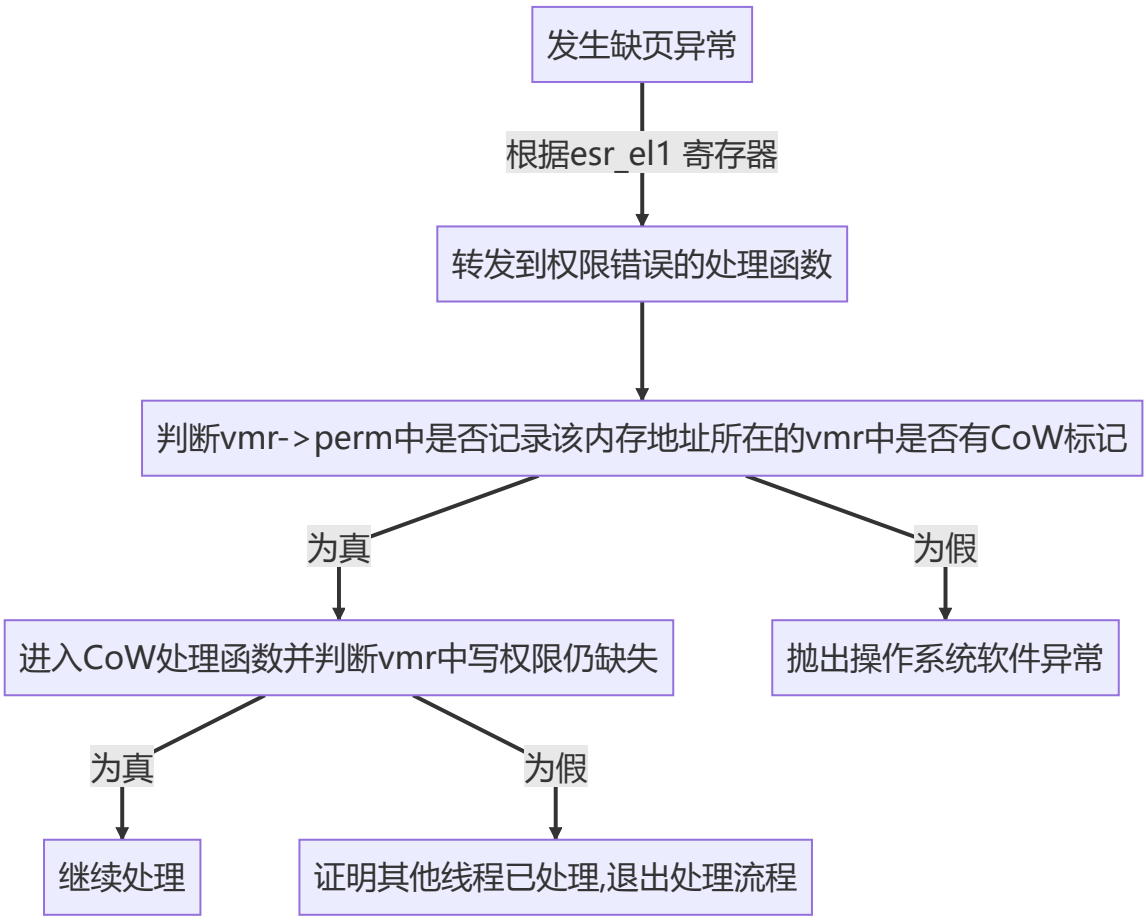
`query_in_pgtbl`、`map_range_in_pgtbl_common`、`unmap_range_in_pgtbl` 和 `mprotect_in_pgtbl` 实现了操作系统对于页表的增删改查功能，使得我们可以操作虚拟地址到物理地址的映射。题目的难点在于将理论的页表维护概念具象化为实现的思路，并结合已给的数据结构进行实现。

详细实现请见源代码

5 思考题5

通过观察 `pgfault.c`,`pgfault_handler.c` 中对COW的处理方式，以及 `pte_t` 结构体规定的页表项的组成方式可知，该页表项的权限为要是 Read-Only，查询 `set_pte_flags` 函数得，即需要页表项的6，7位的Access Permission 为 `2b'11`。

在发生缺页处理时，处理流程如下



其中，**继续处理CoW**的步骤为：

1. 为写时拷贝进行写操作分配新的物理页
2. 在该进程的vmr中记录下写时拷贝所分配的物理页
3. 使用 `memcopy` 将发生缺页异常的地址所在物理页内存，复制到刚分配的物理页中
4. （猜测）在vmr中所有 CoW都因为写时拷贝而消失后，更新 vmr 的权限，取消对 CoW 的支持
5. 在本进程的页表中，更新发生缺页异常的地址的页表项映射，新映射指向新分配且复制好的物理页。
6. 刷新TLB，仅需更新发生缺页异常的地址所在页。

6　思考题6

如果内核页表使用粗粒度映射，那么一整个物理页的大小将会变大。一个虚拟内存区域vmr又由多个物理页组成，因此segment的大小也会变大。如果内核所需要的连续地址空间长度小于2M的范围的话，将用不满一个虚拟内存区域上的所有地址。遇到需要分配多个相互隔离的虚拟内存区域的情况，可能会使得内存空间因为大量内部碎片而不够用。

7 挑战题7

好难

8 练习题8

do_page_fault负责检验esr寄存器的错误信息，并将far中的出错va，所属vmr，权限等信息汇总，然后交给对应过的handler处理函数

详细实现请见源代码

9 练习题9

VMSPACE使用红黑树组织在这个进程中分配好的一个个虚拟内存区域。

我们可以使用rbtree.h中给出的各种操作红黑树的代码，以及vmspace.c中实现的地址比较代码，实现为对应的va在红黑树上查找。实现的关键在于阅读懂所给的代码和红黑树的结构

详细实现请见源代码

10 练习题10

最后我们需要完成按需分配下和合法的缺页异常所对应的处理函数。具体的处理过程跟据PMO的不同分为两种。

如果缺页PA在PMO中没有记录，我们需要

- 分配物理页
- 将分配的物理页pa注册到 PMO 中
- 填写页表

如果PA在PMO中已经记录，有可能是多个线程同时触发了这个缺页异常，我们只需要

- 填写页表
- 不应当重复分配物理页

详细实现请见源代码

11 挑战题11

我太难了