OSlab3--by：周延霖

实验部分

做完实验二后，大家可以了解并掌握物理内存管理中的连续空间分配算法的具体实现以及如 何建立二级页表。本次实验是在实验二的基础上，借助于页表机制和实验一中涉及的中断异 常处理机制，完成Page Fault异常处理和FIFO页替换算法的实现。实验原理最大的区别是在 设计了如何在磁盘上缓存内存页，从而能够支持虚存管理，提供一个比实际物理内存空间“更 大”的虚拟内存空间给系统使用。

实验目的如下：

- 了解虚拟内存的Page Fault异常处理实现
- 了解页替换算法在操作系统中的实现

本次实验是在实验二的基础上，借助于页表机制和实验一中涉及的中断异常处理机制，完成 Page Fault异常处理和FIFO页替换算法的实现，结合磁盘提供的缓存空间，从而能够支持虚 存管理，提供一个比实际物理内存空间“更大”的虚拟内存空间给系统使用。这个实验与实际操 作系统中的实现比较起来要简单，不过需要了解实验一和实验二的具体实现。实际操作系统 系统中的虚拟内存管理设计与实现是相当复杂的，涉及到与进程管理系统、文件系统等的交叉访问。

练习1：给未被映射的地址映射上物理页（需要编程）

问题回答

> - 请描述页目录项（Pag Director Entry）和页表（Page Table Entry）中组成部分对ucore 实现页替换算法的潜在用处。
> - 如果ucore的缺页服务例程在执行过程中访问内存，出现了页访问异常，请问硬件要做哪些事情?

1. 页目录表和mm_struct结构对应，用于根据传入的线性地址索引对应的页表；页表项，即一个PTE用来描述一般意义上的物理页时，应该有PTE_P标记，即表示物理页存在；但当它用来描述一个被置换出去的物理页时，它被用来维护该物理页与swap磁盘上扇区的映射关系，此时没有PTE_P标记。页替换涉及到换入换出，换入时需要将某个虚拟地址对应于磁盘的一页内容读入到内存中，换出时需要将某个虚拟页的内容写到磁盘中的某个位置，因此页表项可以记录该虚拟页在磁盘中的位置，也为换入换出提供磁盘位置信息。
2. CPU会把产生异常的线性地址存储在CR2寄存器中，并且把表示页访问异常类型的error Code保存在中断栈中。

练习2：补充完成基于FIFO的页面替换算法（需要编程）

问题回答

> - 如果要在ucore上实现"extended clock页替换算法"请给你的设计方案，现有的 swap_manager 框架是否足以支持在ucore中实现此算法？如果是，请给你的设计方案。 如果不是，请给出你的新的扩展和基此扩展的设计方案
>   - 需要被换出的页的特征是什么?
>   - 在ucore中如何判断具有这样特征的页?
>   - 何时进行换入和换出操作?

1. 现有的swap_manager框架支持在ucore中实现此算法
2. 具有这样特征的页应该满足如下特征
    - 优先选择 Dirty Bit 为0，Access Bit 为0 的页
    - 其次选择 Dirty Bit 为0，Access Bit 为1 的页
    - 最后选择 Dirty Bit 为1，Access Bit 为0 的页

    转换为代码如下

    ```
    !(*ptep & PTE_A)&&!(*ptep & PTE_D); //最优
    (*ptep & PTE_A) &&!(*ptep & PTE_D); //次优
    !(*ptep & PTE_A)&& (*ptep & PTE_D); //最次
    ```

3. 缺页的时候换入，满页的时候换出

总结

1. 其实练习1比起后面的实现相对简单，主要就是一步步确认这个异常的确是缺页异常。do_pgfault主要是设置了层层if检查；
2. 练习1所涉及的get_pte函数以及pgdir_alloc_page函数在Lab2中都有所接触，所以用起来还算顺利，但我一般不会进行分配失败的检查，后来查看其他代码修改的时候才发现不写NULL检查的确不安全。所以以后还是要养成好的代码习惯。
3. do_pgfault这个函数实现的是扇面的换入，即swap_in，没有swap_out功能的实现；在考察swap_out功能实现的时候，一开始没想到alloc_pages这个lab2已经实现过的函数会被修改，并完成了这么重要的任务，所以没找到swap_out。后来在分析代码逻辑的时候，也就是从原理上看什么时候换出的时候才又转去看页面page的代码，发现这个alloc_pages已经和lab2不一样了，执行了swap_out(check_mm_struct,n,0);