练习1：给未被映射的地址映射上物理页（需要编程）

完成do\_pgfault（mm/vmm.c）函数，给未被映射的地址映射上物理页。设置访问权限 的时候 需要参考页面所在 VMA 的权限，同时需要注意映射物理页时需要操作内存控制 结构所指定 的页表，而不是内核的页表。注意：在LAB3 EXERCISE 1处填写代码。执行

make　qemu

后，如果通过check\_pgfault函数的测试后，会有“check\_pgfault() succeeded!”的输出，表示 练习1基本正确。

请在实验报告中简要说明你的设计实现过程。请回答如下问题：

请描述页目录项（Page Directory Entry）和页表项（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、一开始，通过find\_vma函数进行查找，看管理器中是否存在传入的错误地址addr，如果不存在直接报错。

struct vma\_struct \*vma = find\_vma(mm, addr);

pgfault\_num++;

//If the addr is in the range of a mm's vma?

if (vma == NULL || vma->vm\_start > addr) {

cprintf("not valid addr %x, and can not find it in vma\n", addr);

goto failed;

}

2、接下来进行错误类型检查

//check the error\_code

switch (error\_code & 3) {

default:

/\* error code flag : default is 3 ( W/R=1, P=1): write, present \*/

case 2: /\* error code flag : (W/R=1, P=0): write, not present \*/

if (!(vma->vm\_flags & VM\_WRITE)) {

cprintf("do\_pgfault failed: error code flag = write AND not present, but the addr's vma cannot write\n");

goto failed;

}

break;

case 1: /\* error code flag : (W/R=0, P=1): read, present \*/

cprintf("do\_pgfault failed: error code flag = read AND present\n");

goto failed;

case 0: /\* error code flag : (W/R=0, P=0): read, not present \*/

if (!(vma->vm\_flags & (VM\_READ | VM\_EXEC))) {

cprintf("do\_pgfault failed: error code flag = read AND not present, but the addr's vma cannot read or exec\n");

goto failed;

}

}

第一种：申请写操作，物理内存中不存在，并且对应地址的内容不允许写

第二种：申请读操作，并且物理内存中存在（因此已经报错，所以是权限不够）

第三种：申请读操作但是物理内存中不存在，并且该地址数据不允许被读或者加载

排除这三种情况就是操作系统需要处理的情况

包括 写一个存在的地址 写一个不存在并且addr是可写的 读一个不存在地址并且addr是可读的

这时候我们就要进行相应的处理了，以便程序能够正常运行

1、首先去查找addr对应的PTD（二级页表）条目，如果不存在则分配一页，若分配出错，则报错

// try to find a pte, if pte's PT(Page Table) isn't existed, then create a PT.

// (notice the 3th parameter '1')

if ((ptep = get\_pte(mm->pgdir, addr, 1)) == NULL) {

cprintf("get\_pte in do\_pgfault failed\n");

goto failed;

}

2、如果是新分配的PTD,那么相应的二级页表条目应该是空的(\*ptep=0)，这时候我们就分配一页物理空间，建立映射关系，如果分配失败，则报错

if (\*ptep == 0) { // if the phy addr isn't exist, then alloc a page & map the phy addr with logical addr

if (pgdir\_alloc\_page(mm->pgdir, addr, perm) == NULL) {

cprintf("pgdir\_alloc\_page in do\_pgfault failed\n");

goto failed;

}

}

练习2：补充完成基于FIFO的页面替换算法

剩下的就是信息被存放到磁盘上去的，这时候我们需要将这页信息从磁盘上交换回来

else { // if this pte is a swap entry, then load data from disk to a page with phy addr

// and call page\_insert to map the phy addr with logical addr

if(swap\_init\_ok) {

struct Page \*page=NULL;

if ((ret = swap\_in(mm, addr, &page)) != 0) {

cprintf("swap\_in in do\_pgfault failed\n");

goto failed;

}

page\_insert(mm->pgdir, page, addr, perm);

swap\_map\_swappable(mm, addr, page, 1);

page->pra\_vaddr = addr;

}

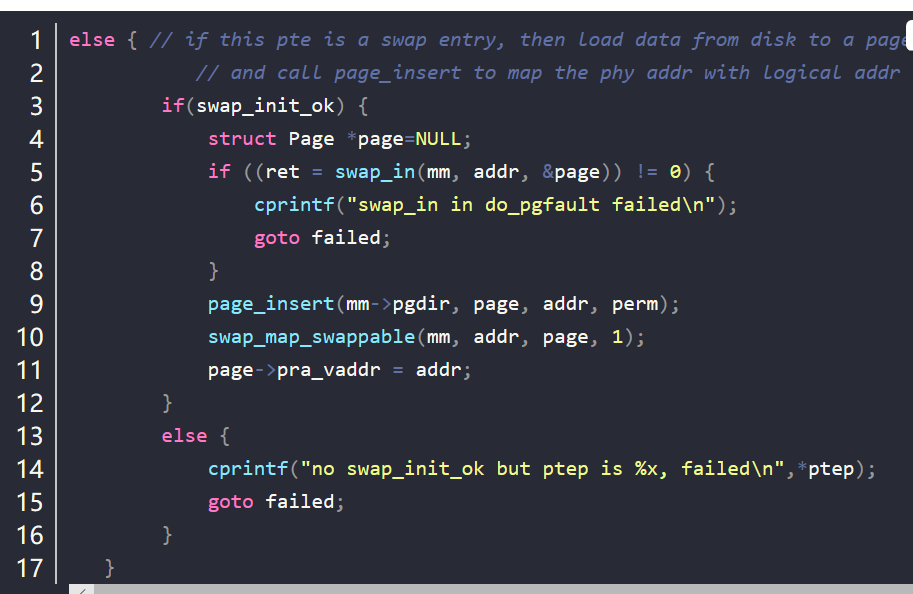
else {

cprintf("no swap\_init\_ok but ptep is %x, failed\n",\*ptep);

goto failed;

}

}



1、首先确认交换初始化完毕 然后分配一页物理空间，利用swap\_in函数，将磁盘上的信息交换到物理内存中

2、 然后利用page\_insert函数将新申请的物理页面与线性地址建立映射关系

3、最后利用swap\_map\_swappable函数根据FIFO替换规则，将最近到达的页面链接到队列的最后对上述出现的函数进行说明：

swap\_in: 根据PTE中的addr交换项，得到对应在磁盘中的位置，读取相应位置的信息到内存页中

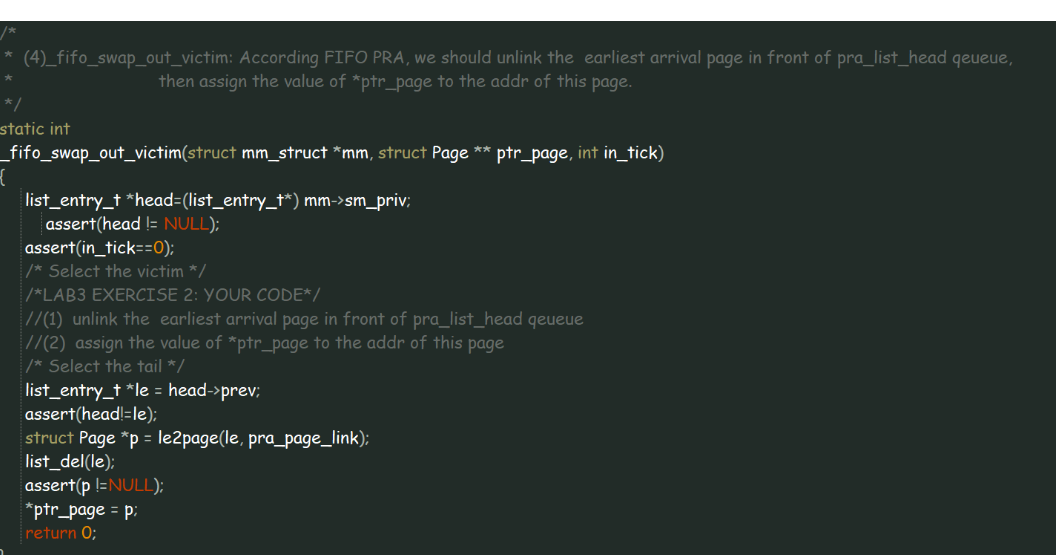
page\_insert:将新申请的物理页面与线性地址建立映射关系

原型：\_fifo\_map\_swappable

/mm/swap\_fifo.c

swap\_map\_swappable：根据FIFO替换规则，将最近到达的页面链接到队列的最后

最后的一步工作便是将swap\_fifo.c文件中的——fifo\_swap\_out\_victim函数填充完整，这个函数的作用是在进行页替换时选择哪一页进行替换，由于我们采用的FIFO的方法，因此，只要取出队列的第一个进行替换就好了



最后利用 make qemu进行结果查看，出现了

check\_pgfault() succeeded!

check\_swap() succeeded!

的字样，结果正确