****



**MIT xv6实验报告**

**——lab6 : cow**

**学生姓名 胡轶然**

**学 号 3019244355**

1. **实验目的**
2. 实现copy on write。
3. 理解虚拟内存的工作原理。
4. 熟悉xv6的内存分配机制。
5. **前期准备**
6. 切换git分支。
7. 阅读指导书，了解与虚拟内存和内存管理相关的知识。
8. 阅读kernel文件夹下的部分代码。
9. **实验内容及实现步骤**

**任务1：Implement copy-on write**

1. **问题描述**

xv6系统在调用fork函数创建子进程时，会将父进程持有的物理内存复制一份分配给子进程。然而，这种做法存在着弊端：通常情况下，子进程被创建后会清空原有内存空间并执行exec函数，而复制和释放这部分内存的操作会浪费一定的时间和内存空间。

在本实验中，我们尝试通过copy-on write技术解决上述问题。Copy-on write技术类似于lazy allocation，在创建子进程时，不会立刻为子进程复制父进程的物理内存，而是将子进程的虚拟内存关联到父进程的物理内存。

1. **思路与实现步骤**

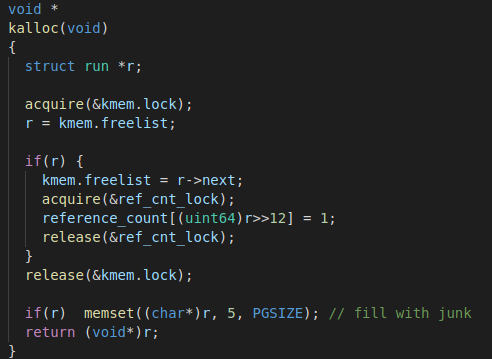
为了在实现copy-on write后仍能保证进程间内存的**隔离性**，为物理内存的每一页都设置一个特殊的标记位(cow)和引用计数器。

在执行fork操作时，不复制父进程的物理内存，而是将子进程的虚拟内存关联到父进程的物理内存。同时取消父、子进程对共用物理内存的写权限。

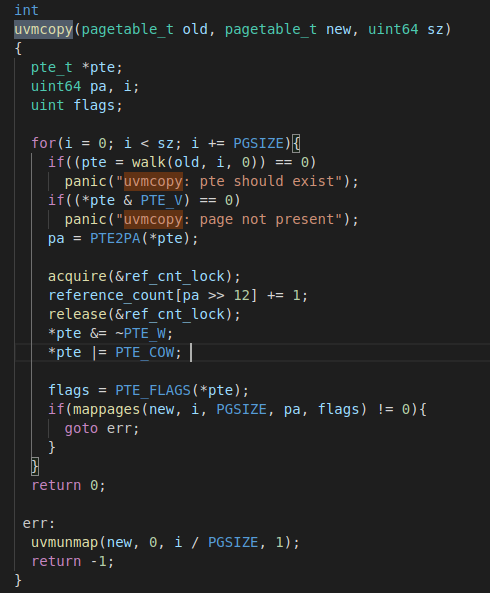
修改缺页逻辑，当有进程试图修改持有cow标记的内存时，系统会为该进程分配一份目标内存的拷贝以供操作。当有程序试图释放引用数大于1的内存时，系统会阻止释放操作，并将引用数减1。

具体思路和实现步骤如下：

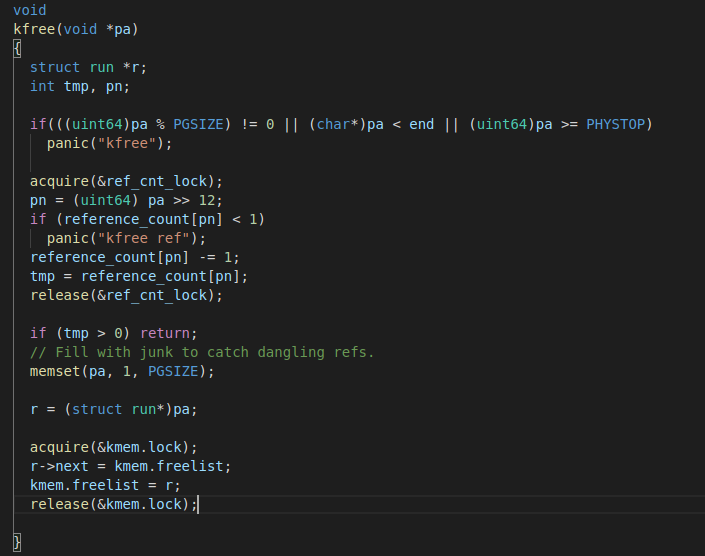
1. 添加一个新的PTE权限：COW，用于标记多个进程共用的物理内存页。
2. 创建一个数组，为每一页物理内存都设置一个引用计数器，同时，添加一个自旋锁以防止多个进程同时操作计数器。注意到**物理页大小为4096=1012字节，故计数器数组长度为PHYSTOP （物理内存大小）>> 12**。
3. 在kinit函数中，初始化上文添加的自旋锁。
4. 在kalloc函数中，将分配给进程的物理内存页的引用计数初始化为1。



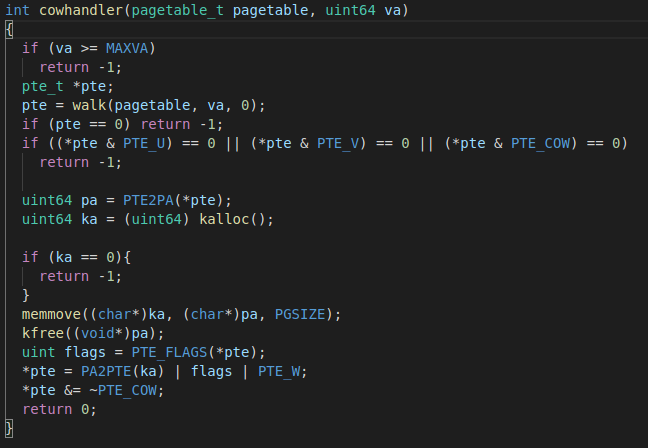
1. 修改fork函数调用的uvmcopy函数，具体内容如下：
   1. 不再为子进程复制父进程的物理内存，删去对kalloc函数的调用。
   2. 增大共享物理块的引用计数，并关闭共享物理块的写权限，打开其COW权限。



1. 修改kfree函数中，释放内存页的逻辑，具体内容如下
   1. 通过将内存地址右移12位得到内存页号，并使用页号获取内存页的引用计数。
   2. 若引用计数小于1，程序发生异常，调用panic。
   3. 若引用计数不小于1，将引用计数减1.
      1. 若减1后引用计数等于0，代表没有进程持有该内存页，此时可以释放该内存页。
      2. 若减1后引用计数大于0，代表仍有程序持有该内存页，此时不能释放该内存页，直接返回。



1. 修改usertrap函数中，处理缺页异常的逻辑。此处的修改类似Lab 5(lazy allocation)，添加了一个判定、处理copy-on write现象的函数，当发生回写异常（异常编号15）时，并在发生内存写异常时调用该函数，并根据其返回值判断copy-on write是否成功。函数的具体内容如下
   1. 使用walk函数获取内存页的pte（物理地址+权限码），并判断权限是否符合copy-on write的要求。
   2. 使用PTE2PA获取内存页的物理地址，然后使用kalloc和memmove函数将该物理页复制一份，供执行写操作的进程使用。
   3. 调用kfree函数，降低原物理页的引用数。
   4. 打开新物理页的写权限，并关闭其COW权限。





1. **问题与解决方法**

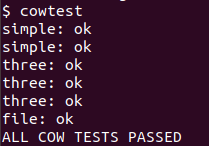
**问题一**：最初，编写的引用计数逻辑总是存在问题，常出现一些奇怪的引用值和错误。

**解决方法**：每次修改引用计数代码后，都打印引用计数的值和做出的修改，**发现将引用计数错误地初始化为0，而不是1。**

**问题二**：解决引用计数的问题后， 发现存在缺页错误。

**解决方法**：经检查和讨论，发现**在cowhandler函数复制了共享物理页后，没有修改新物理页的权限，导致新物理页也是只读的**，当进程对新物理页执行写操作时就会发生错误。

1. **结果**



1. **实验结果**

Make grade

