哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验设计报告

开课学期：2022 年秋季

课程名称：操作系统

实验名称：COW

实验性质：额外实验

实验时间：10.1 地点：T2

学生班级：11

学生学号：200111132

学生姓名：吴桐

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2022 年 9 月

一、 实验详细设计

Cow 实验（奶牛实验）

实验的目的在于，当我们想要 fork 的时候，对于一块物理内存而言，如果他是只读（后续）的话，那么其实无需复制物理内存。当我们使用这同一块物理内存的进程组中，出现了需要修改该块内存的时候，我们才选择单独分配和复制一块新的物理内存供他使用。

首先我们要考虑内存的分配问题。对于此时被共享的页我们给他一个特殊的标记 COW 标记（PTE tag）。因为本质还是存在一个进程的页表当中，只是指向同一个物理区域。这里主要是修改 uvmcopy 函数，不 kalloc，但父子 pte 都需要加上 cow 位。以及将 PTE_W 置为 0，当我们需要操作这一页的时候，进入 trap 里面再处理。

然后我们会想到一个实际物理页可能会有多个 cow PTE。所以我们不难想到要对物理上的每一个页的引用要进行计数（有多少个项指向了它）。同时考虑到多线程我们需要把这个引用计数分别加锁（在我的实现中每一个页都各自有锁和计数，没有 hash（不考虑开销，笑））

然后很多操作就是和 refcnt 相关的分配和修改内存（肯定会包括锁的一些操作啦，略）：

Kalloc 的时候，将引用计数设置为 1

Kfree 的时候，判断一下引用计数是否为 1，如果不为 1，则只用 refcnt--。当 refcnt==1 的时候我们才释放实际的物理页。

最后就是我们的有关如何写和实际分配的函数 cowcopy 写在 kalloc.c 。
Trap 里面当发生写异常的时候 (r_scause()==15) 就进行 cowcopy 的操作：

先看看是不是 cow 页发生了异常，否则就是正常写异常，需要出错。

如果是 cow 页的话，先看 refcnt：

如果 refcnt>1，则是 cow 页中的一个，我们为其复制内容到新的一个物理页，取消 cow 位并且 pte_w 置为 1。

如果 refcnt 为 1，则它是最后剩下的 cow 页。我们直接把它 cow 取消，pte_w 为 1 就好了。

当然还有一些设置工具函数的细枝末节操作啦。

二、 实验结果截图

请填写

```
$ make qemu-gdb
(13.6s)
== Test   simple ==
  simple: OK
== Test   three ==
  three: OK
== Test   file ==
  file: OK
== Test usertests ==
$ make qemu-gdb
(198.3s)
== Test   usertests: copyin ==
  usertests: copyin: OK
== Test   usertests: copyout ==
  usertests: copyout: OK
== Test   usertests: all tests ==
  usertests: all tests: OK
== Test time ==
time: OK
Score: 110/110
root@VM-8-8-ubuntu:~/myxv6/github/MIT6.S081-2020-labs#
```