**操 作 系 统**

**实 验 报 告**

**实验名称：实验四 同步互斥问题**

**姓名： 王晶**

**学号： 16340217**

实验名称：同步互斥问题

一、实验目的：

利用线程同步机制，实现生产者-消费者问题和读者-写者问题。

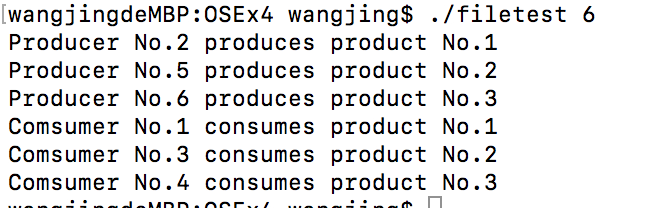
二、实验要求：

通过文件操作，读入数据，并通过信号量，进行对缓冲操作的保护。完成生产者-消费者问题和读者-写者问题。

三、实验过程：

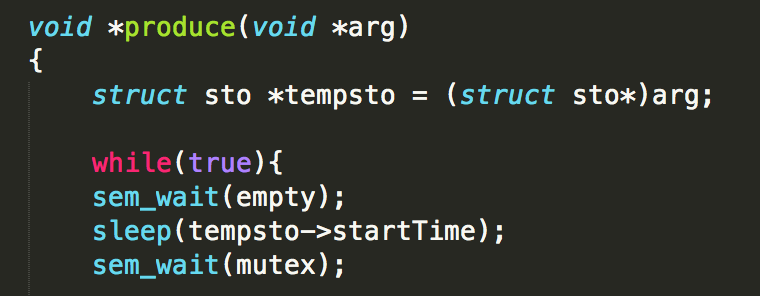
实验环境是在macOS下，基本实现机制和内容和Linux一致，但macOS下支持创建有名的信号量，因此直接使用sem\_init会出现错误。此时需要用到sem\_open函数进行创建，然后通过sem\_unlink进行删除，否则将一直存在知道内核重启。

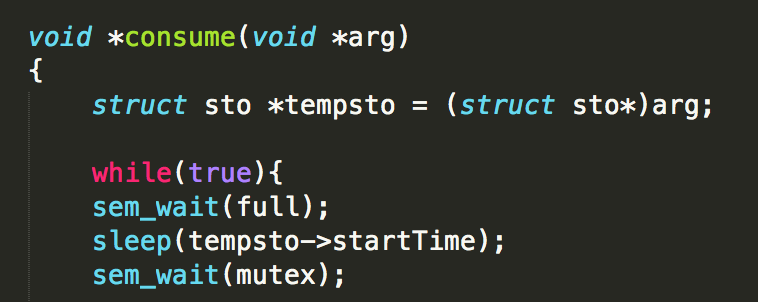
生产者-消费者问题的运行结果如下：



之所以三个生产者先执行，是因为第一个生产者结束时，其他生产者都在等待mutex信号量，但消费者还在等待full信号量，执行速度上有区别。

其中信号量empty用于记录空位，full用于记录满位，mutex用于互斥。生产者先等待empty信号量，然后等待mutex信号量进行操作。而消费者先等待full信号量，再等待mutex信号量进行操作。





读者-写者问题：

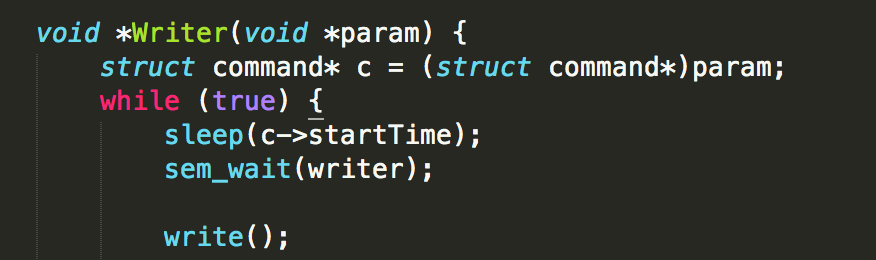
1. 读者优先：

该情况下，多个读者可以同时访问资源，因此不需要设置信号量来限制读者之间对资源的访问，读者和写者之间互斥，而写者之间也是互斥的。

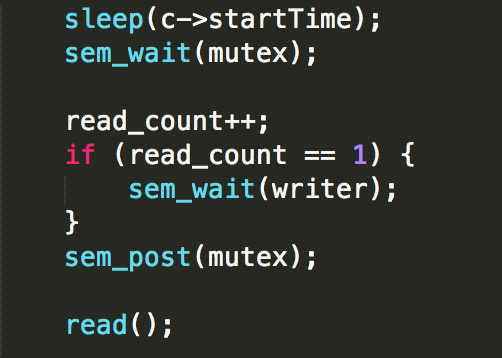
其中mutex用于写者之间互斥，writer用于读者写者之间互斥。

除此之外read\_count用于记录当前读者的数量，当有读者进入时锁住信号量writer，直到所有读者线程结束，才释放writer。

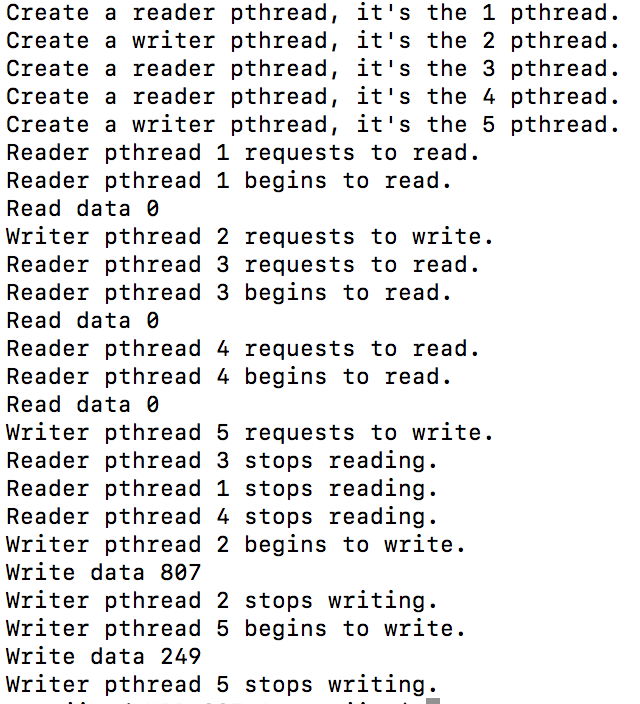
写者只需等待write信号量：



而读者要等待mutex信号量，并且对write信号量进行操作从而阻塞写者：



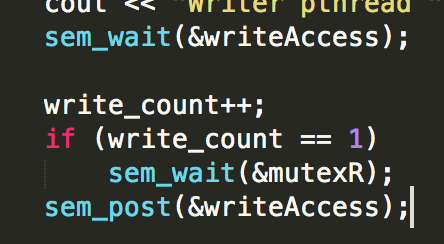
结果如下：



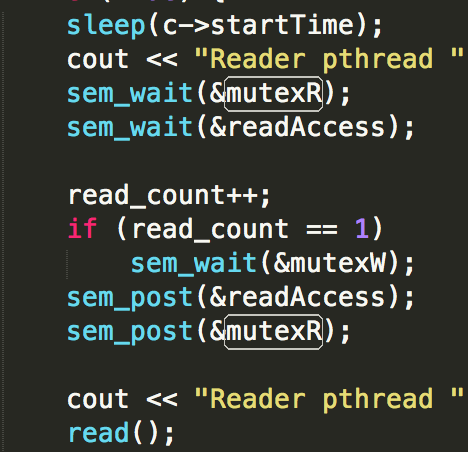
1. 写者优先：

在该情况下，写者在等待，则新来的读者将不允许进行操作，进入等待，这时用变量write\_count记录写者的数量，等于0，即所有写者结束时释放读者的线程。通过信号量mutexR在写者线程中阻塞读者线程，信号量mutexW在读者线程中阻塞写者线程。ReadAccess对read\_count实现互斥，WriteAccess对write\_count实现互斥。

通过writeAccess对write\_count进行保护，read\_count同理。并且有写者排队时，通过mutexR阻塞读者。



写者线程可以畅通无阻的从开始运行到等待操作mutexR，即阻塞读者线程；而读者线程一开始就要等待mutexR，这样就是实现了当写者进入等待时，除了正在运行的读者，其他读者都要进入等待，直到写者完成。



运行结果如下：

