**实验三：进程管理设计报告（阐述设计程序的思路）**

**writer.c:**

首先在writer.c中定义好需要的头文件如编译c语言的stdio.h，stdlib.h以及POSIX特有的pthread.h，unistd.h，semaphore.h。之后宏定义写者读者数量以及读者写者访问共享区域的频率（即时间间隔）。然后就是定义读者写者序列，数据生成范围，当前读者写者数量以及需要的各种锁变量。这里着重提一下锁变量的设计：accessReaderCnt用来保证有读者读时写者不允许写，同时还要设计成可以保证多个读者读的情况；accessWriterCnt保证写者之间互斥和读者写者的互斥；writeLock和readerLock保证写者和读者在访问临界区资源时互斥；最重要的设计是outerLock，它保证了写者优先。

write1和read1函数没什么好说的，就是写者和读者访问临界区资源的过程（这里访问的是全局变量data）。writer函数首先访问锁变量accessWriterCnt，希望修改writerCnt，独占writerCnt，如果已经有写者了则当前写者被阻塞。若访问成功则自增writerCnt的值，并判断是否为第一个写者，若是则还需要锁上readerLock，阻止后续的读者加入待读队列，保证写者优先，然后释放锁accessWriterCnt。接着就是写者访问临界区，访问前加锁writerLock，访问完毕解锁writerLock。最后这个写者进程完毕还要自减writerCnt，并判断若是最后一个写者，则释放锁accessWriterCnt，允许后续的读者加入待读队列。reader函数首先访问锁outerLock，如果写者锁定了readerLock，那么读者在这里就会被堵塞（保证写者和读者在队列中争夺访问临界区资源时写者优先），若成功，则读者加锁readerLock，保证此时修改readerCnt的过程只被一个读者访问，并且判断如果是第一个读者，则需要加锁writerLock阻塞写者，之后释放锁readerLock。然后读者进行读操作，最后释放掉readerCnt，并判断若是最后一个读者则解锁writerLock，允许写者执行写进程。

main函数种简单为每个读者和写者创建各自的线程，然后就是观察屏幕上的输出即可。

**thread.c:**

thread.c文件其实整体思想跟上面的writer.c接近，区别在于thread里面没有任何变量用来维护访问临界区资源。定义了共享数据data，缓冲区种元素个数cnt（虽然好像后面并没有使用），生产者及消费者进程序列pid和cid。produce\_item函数和remove\_item函数分别对共享数据进行写操作和读操作，然后producer函数和consumer函数跟上面的writer和reader思想基本一致，只是将其中加锁和解锁的部分删除了。main函数中就创建出producer和consumer进程对临界区进行访问操作。