**操 作 系 统**

**实 验 报 告**

**实验名称：实验三 同步互斥问题**

实验名称：同步互斥问题

一、实验目的：

熟悉线程之间的同步、互斥机制，能通过设计程序模拟线程之间的同步及互斥。

二、实验要求：

1.Linux下程序编译和调试实验

2.编写makefile实验

三、实验过程：

1.生产者-消费者问题

要求：

利用线程同步机制，实现生产者-消费者问题。设计一个程序来解决有限缓冲问题，其中的生产者与消费者进程如图6.10 与图6.11 所示。

在6.6.1 小节中，使用了三个信号量: empty (以记录有多少空位)、full (以记录有多少满位)以及mutex (二进制信号量或互斥信号量，以保护对缓冲插入与删除的操作)。对于本项目， empty 与full 将采用标准计数信号量，而mutex 将采用二进制信号量。生产者与消费者作为独立线程，在empty、full、mutex 的同步前提下，对缓冲进行插入与删除。本项目，可采用Pthread 。

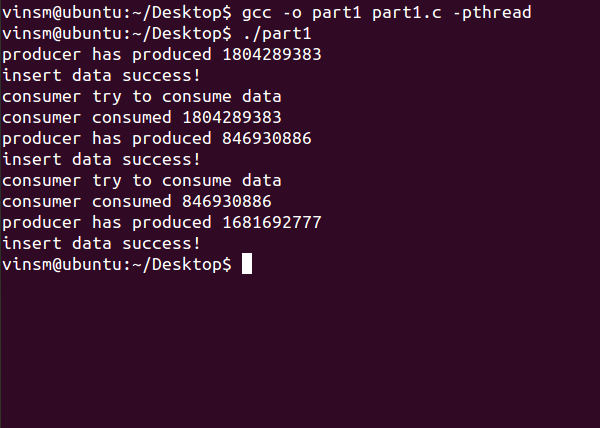
测试数据文件包括n行测试数据，分别描述创建的n个线程是生产者还是消费者，以及生产者或消费者存放或取产品的开始时间和持续时间。每行测试数据包括四个字段，各个字段间用空格分隔。第一字段为一个正整数，表示线程序号。第二字段表示相应线程角色，P表示生产者，C表示消费者。第三字段为一个正数，表示存放或取出操作的开始时间：线程创建后，延迟相应时间(单位为秒)后发出对共享资源的使用申请。第四字段为一个正数，表示操作的持续时间。第五字段为一个正数（仅生产者有），表示生产的产品号。当线程申请成功后，开始对共享资源的操作，该操作持续相应时间后结束，并释放共享资源。

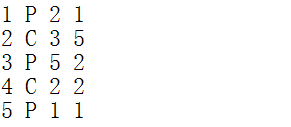
分析：

对于每个生产者线程，需要产生一个随机数，并将它插入到有界缓冲区中；对于每个消费者线程，需要从有界缓冲区中取出一个数据并将它从缓冲区中删除。生产者和消费者线程各自的操作抽象为2个函数，分别为insert\_item和remove\_item。在主函数中创建线程，依据是所读取的测试文件中的内容。根据课本上的图例和程序框架进行设计。

为了保证线程之间不互相抢夺资源，导致插入、删除出错，使用POSIX信号量进行同步和互斥。用POSIX 信号量可以实现传统操作系统P、V操作(即对应课本的wait、signal)。由于POSIX信号量不是内核负责维护，所以当进程退出后，POSIX信号量自动消亡。其中sem\_post()相当于signal操作，释放资源；sem\_wait()相当于wait操作，即申请资源。

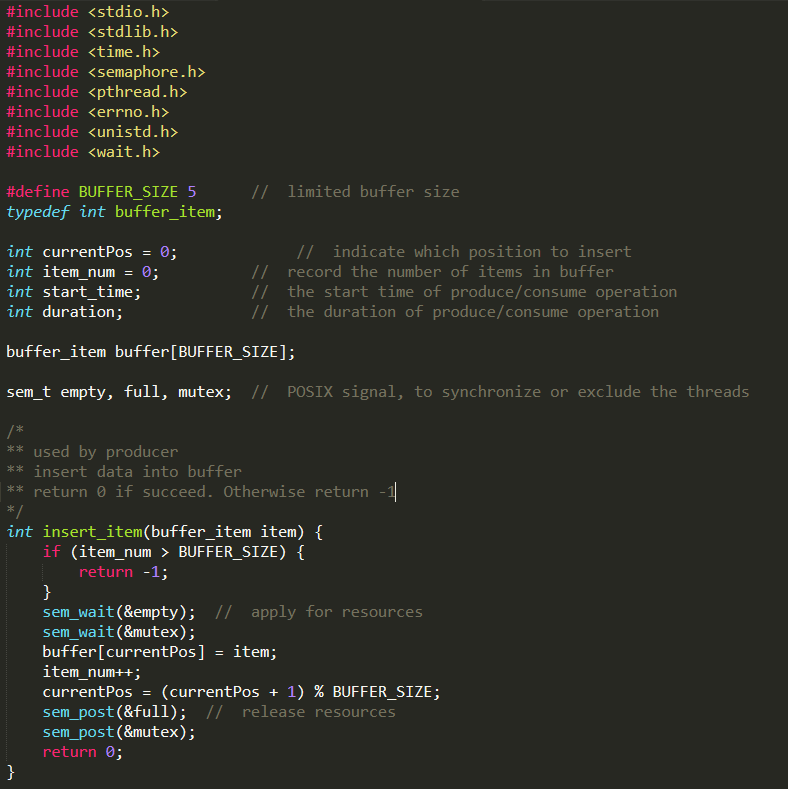
运行结果：

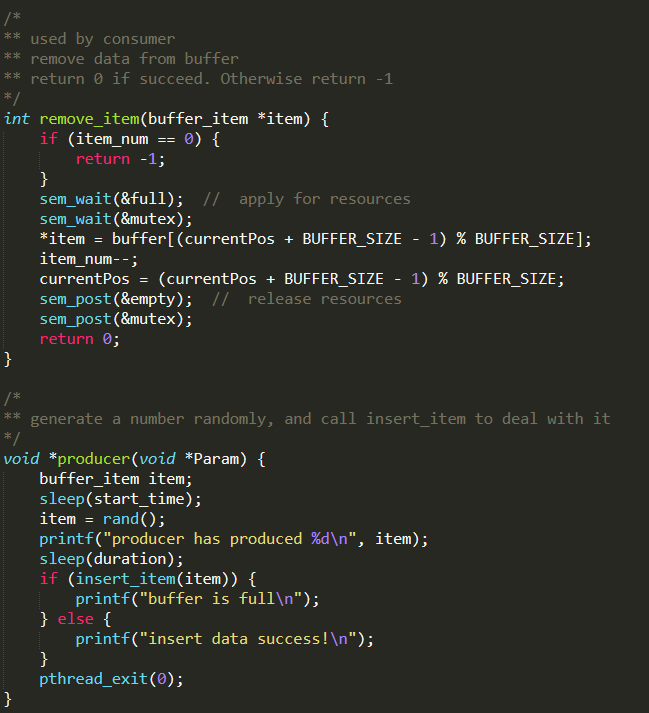




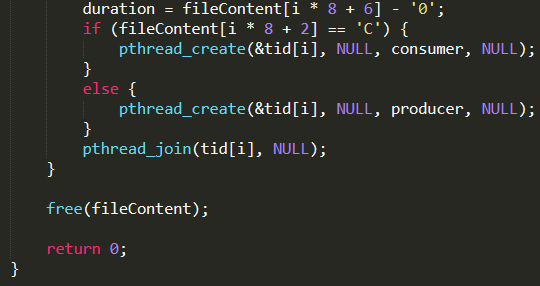
根据测试文件，第一个是生产者线程，存放的开始时间为2，操作持续时间为1，接着是消费者线程、生产者线程、消费者线程、生产者线程，存放或取出时间以及操作持续时间的判断同前。由结果可以看到，1号线程产生了一个随机数1804289383并存进缓冲区，2号线程取出了它，3号线程产生了随机数846930886并存进缓冲区，4号线程取出了它，最后5号线程产生了一个随机数并存进缓冲区，线程之间无冲突，与预设相符。

程序代码：









2.读者写者问题

要求：

在Linux环境下，创建一个进程，此进程包含n个线程。用这n个线程来表示n个读者或写者。每个线程按相应测试数据文件(后面有介绍)的要求进行读写操作。用信号量机制分别实现读者优先和写者优先的读者-写者问题。

读者-写者问题的读写操作限制(仅读者优先或写者优先)：

1)写-写互斥，即不能有两个写者同时进行写操作。

2)读-写互斥，即不能同时有一个线程在读，而另一个线程在写。

3)读-读允许，即可以有一个或多个读者在读。

读者优先的附加限制：如果一个读者申请进行读操作时已有另一个读者正在进行读操作，则该读者可直接开始读操作。

写者优先的附加限制：如果一个读者申请进行读操作时已有另一写者在等待访问共享资源，则该读者必须等到没有写者处于等待状态后才能开始读操作。

运行结果显示要求：要求在每个线程创建、发出读写操作申请、开始读写操作和结束读写操作时分别显示一行提示信息，以确定所有处理都遵守相应的读写操作限制。

测试数据文件包括n行测试数据，分别描述创建的n个线程是读者还是写者，以及读写操作的开始时间和持续时间。每行测试数据包括四个字段，各个字段间用空格分隔。第一字段为一个正整数，表示线程序号。第二字段表示相应线程角色，R表示读者，W表示写者。第三字段为一个正数，表示读写操作的开始时间：线程创建后，延迟相应时间(单位为秒)后发出对共享资源的读写申请。第四字段为一个正数，表示读写操作的持续时间。当线程读写申请成功后，开始对共享资源的读写操作，该操作持续相应时间后结束，并释放共享资源。

2-1 读者优先

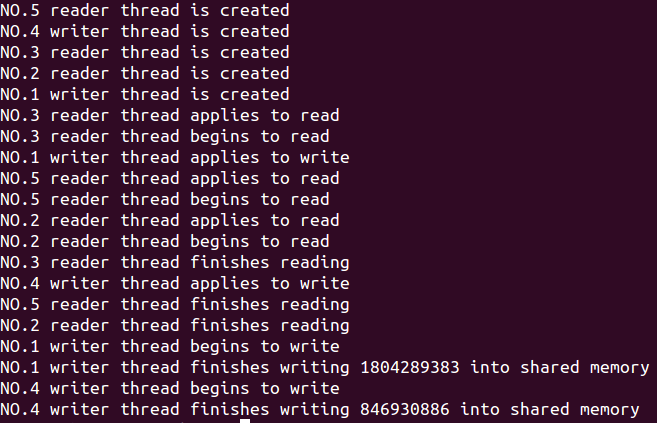
分析：

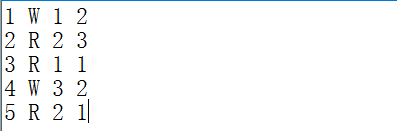
读者优先的几种情况，就是当临界区已经有读者，其他读者可以进入临界区；写者必须等待所有读者完成后，才能进入临界区，即临界区为空时写者才能进入；临界区存在写者时，读者不能进入，其他写者也不能进入。

使用read\_count变量记录临界区中读者数目，第一个进入临界区的读者除mutex外还将申请write\_sig资源，这就阻塞了后来的写者线程，若已有写者申请了write\_sig资源，则读者必须等待。再用mutex在修改read\_count变量值得临界区进行互斥，保证一次只有一个线程改变量的值。

根据测试文件数据，创建读者、写者线程，用结构体给它们传参数。

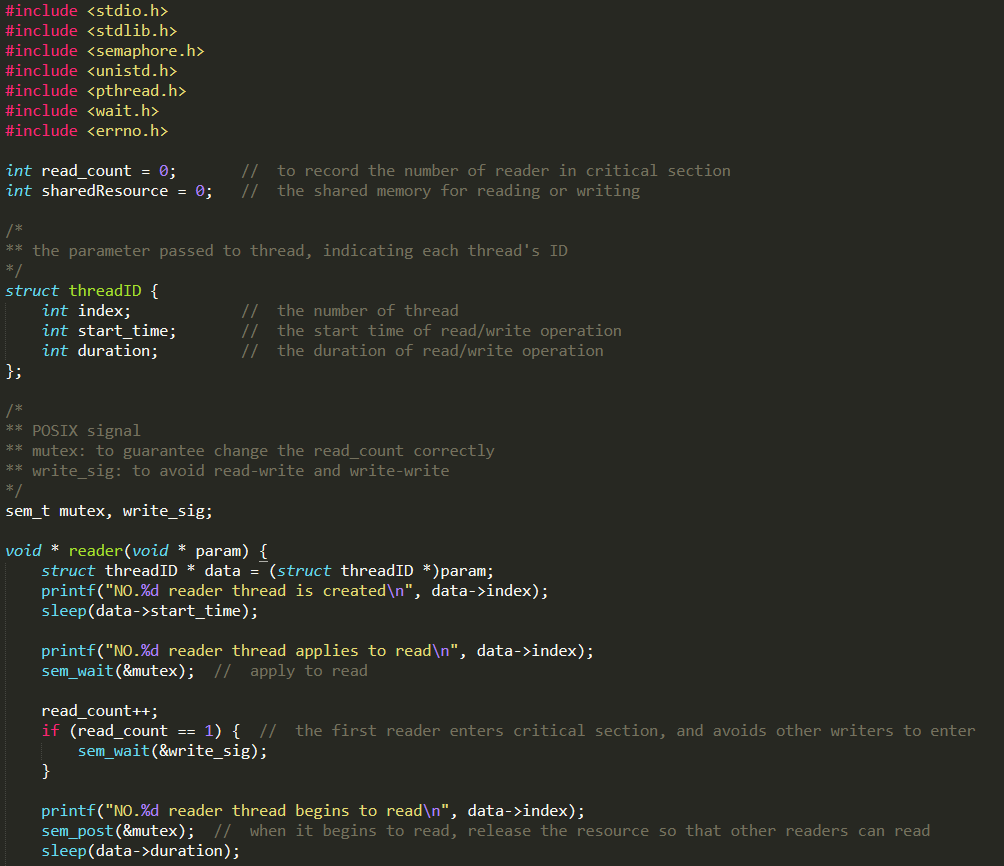
运行结果：

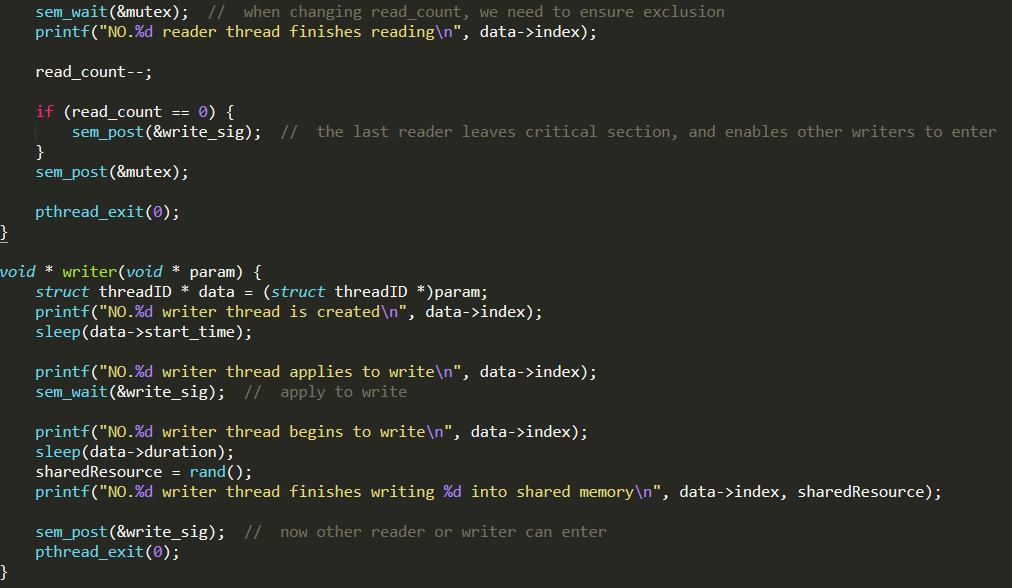


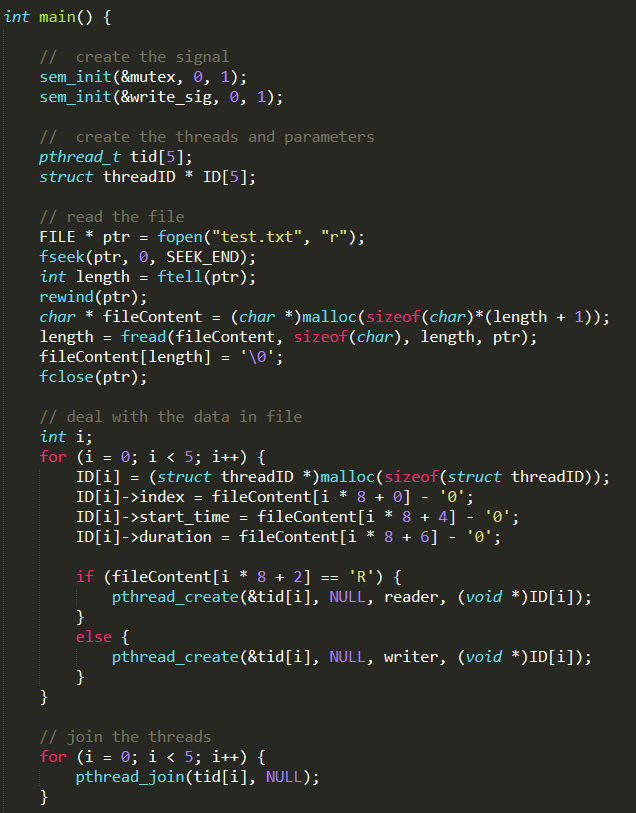


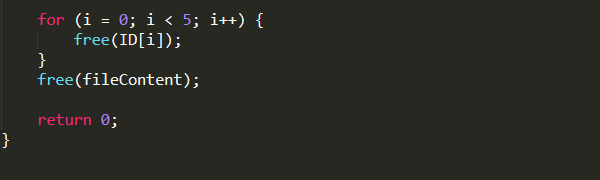
从结果可以看到，5个线程首先被创建，3号读者首先申请进入临界区，虽然接着1号写者申请进入，但它被阻塞，而后面的读者并没有被阻塞。当临界区中所有的读者结束读操作后，1号写者和4号写者才能分别先后进入，与预设相符。

程序代码：









2-2 写者优先

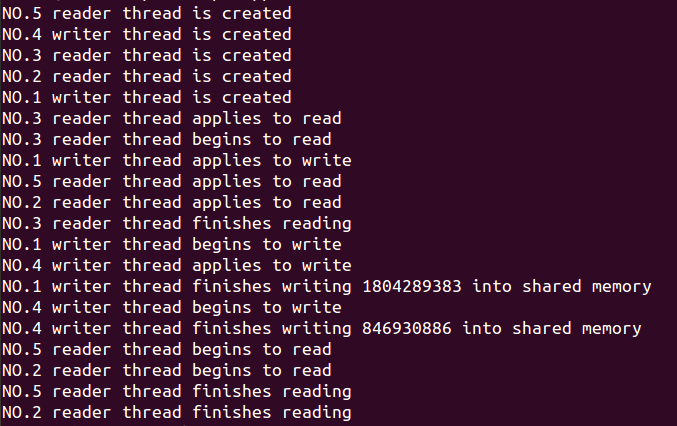
分析：

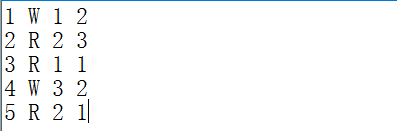
对于写者优先，其实可以在读者优先的基础上面改。读者优先是当临界区有读者，后来的读者就一定可以进入，而写者优先则不然，要根据后来的读者前面有没有写者在等待，如果有，读者就被阻塞了，直到写者完成写操作。

因此，需要5个信号量。change\_rc和change\_wc用来保证改变read\_count和write\_count时互斥；read\_sig是在写者线程等待时申请，使读者线程被阻塞；write\_sig用于避免读写冲突和写写冲突，当读者进程进入临界区时申请，使写者被阻塞；read\_mutex对阻塞read\_sig这一过程进行阻塞。

根据测试文件数据，创建读者、写者线程，用结构体给它们传参数。

运行结果：





从结果可以看到，读者与写者互斥，写者之间互斥，当1号写者在等待时，后面的5、2号读者都不能进行读操作，实现了写者优先，与预设相符。

程序代码：

