**操作系统课程设计 实验报告**

姓名：刘天祺

班级：07121502

学号：1320151097

学院：计算机学院

专业：物联网工程

日期：2018年4月7日

**实验三　生产者消费者问题**

1. 实验要求

通过编写多进程程序实现典型的生产者和消费者问题。实验要求如下：

• 完成 Windows 版本和 Linux 版本

• 一个大小为 3 的缓冲区，初始为空

• 2 个生产者

– 随机等待一段时间，往缓冲区添加数据，

– 若缓冲区已满，等待消费者取走数据后再添加

– 重复 6 次

• 3 个消费者

– 随机等待一段时间，从缓冲区读取数据

– 若缓冲区为空，等待生产者添加数据后再读取

– 重复 4 次

• 显示每次添加和读取数据的时间及缓冲区的状态

1. 实验环境

2.1 Linux环境

操作系统：Ubuntu 14.04.5 LTS 64bit

Shell：zsh 5.0.5 (x86\_64-pc-linux-gnu)

编译器：gcc 4.8.5 (Ubuntu 4.8.5-2ubuntu1~14.04.1)

2.2 Windows环境

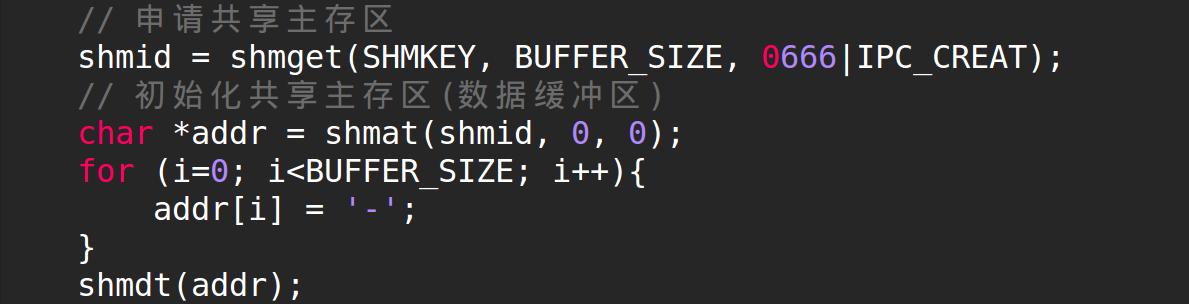
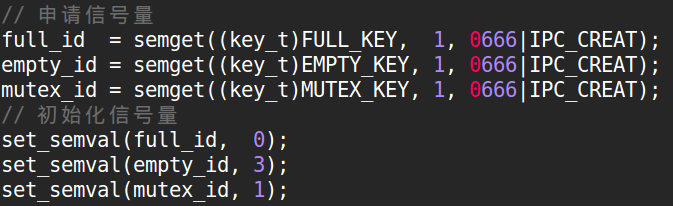
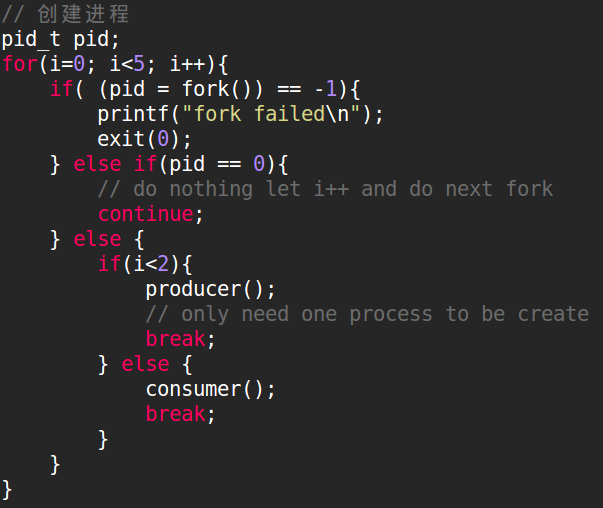
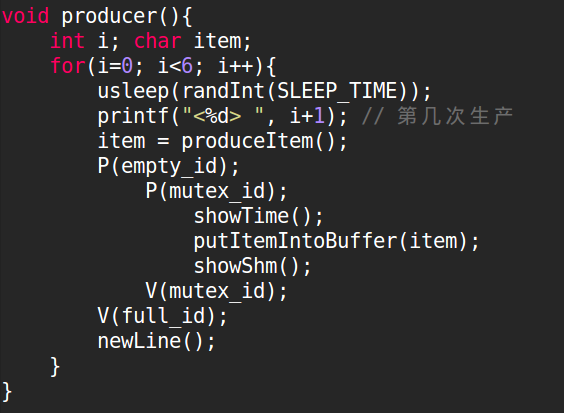
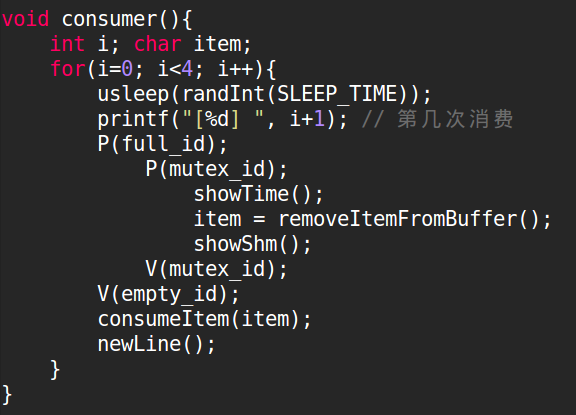
操作系统：Windows 10 64bit

Shell：cmder 160710

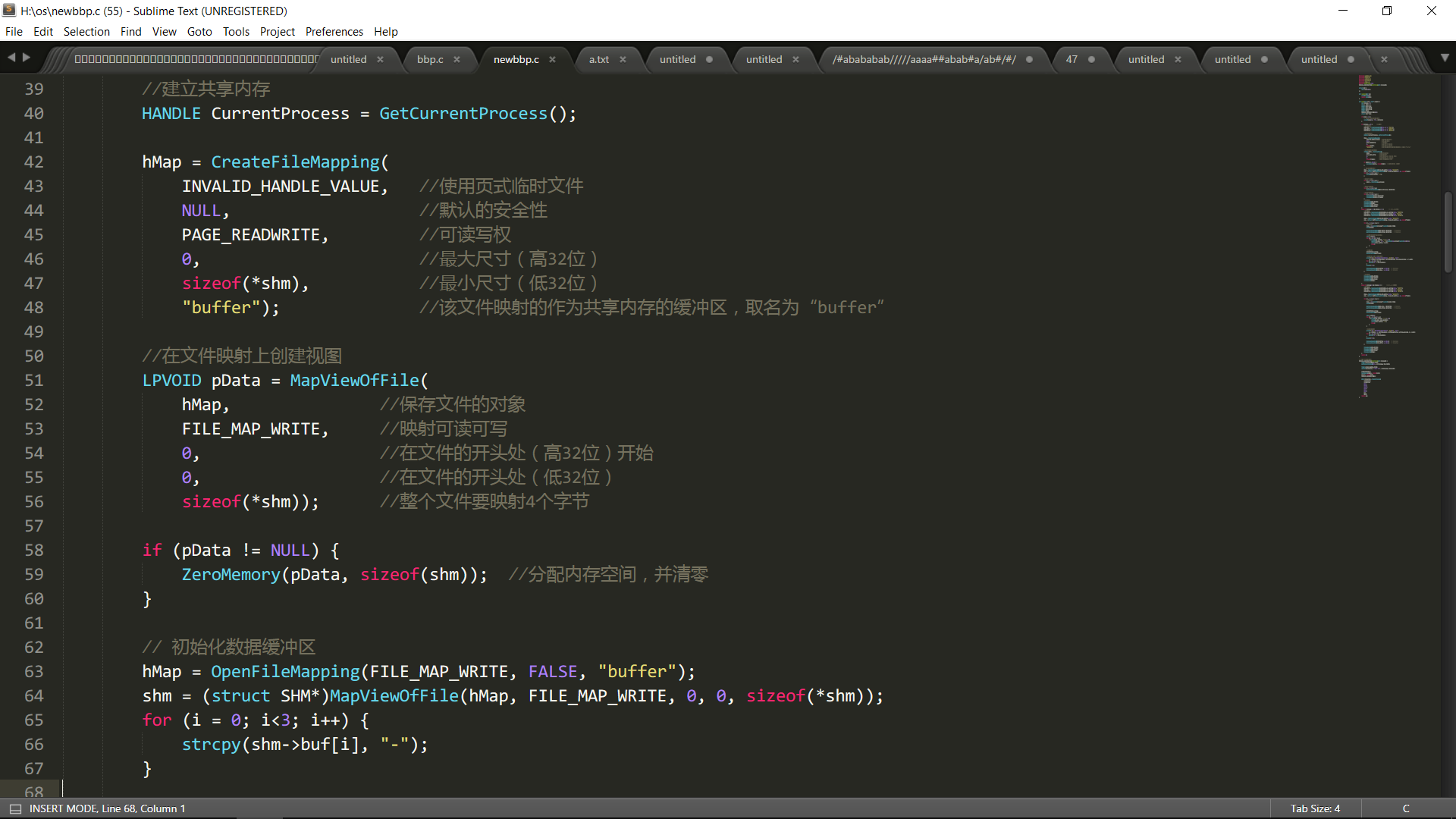
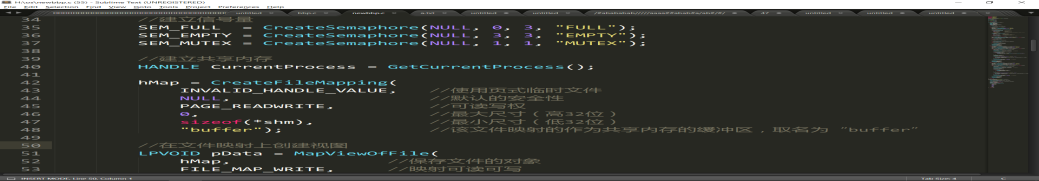
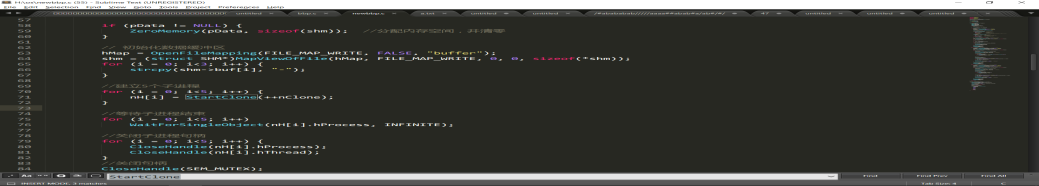
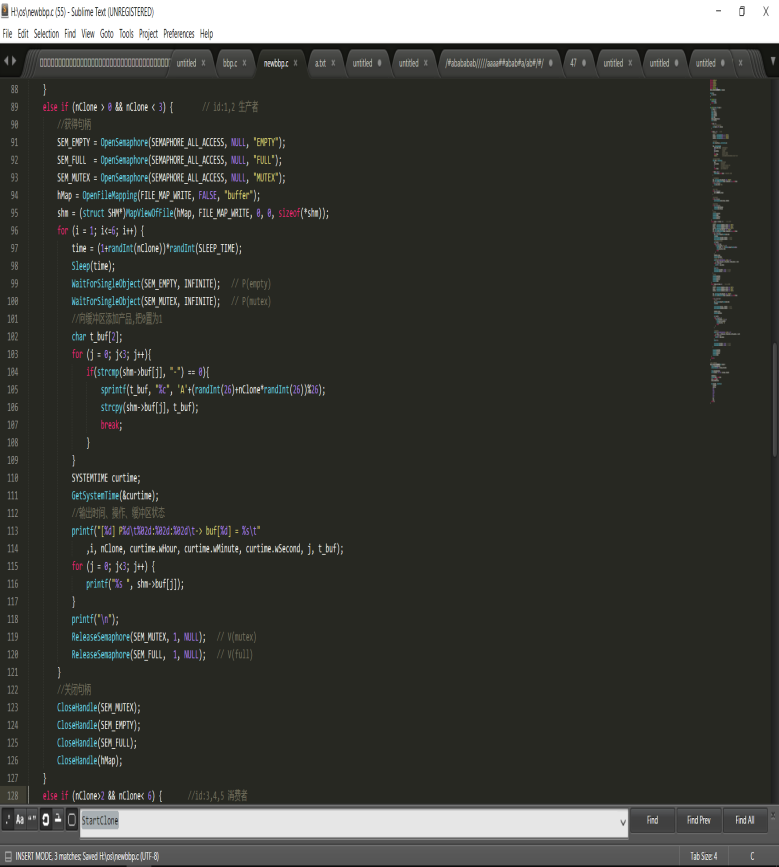
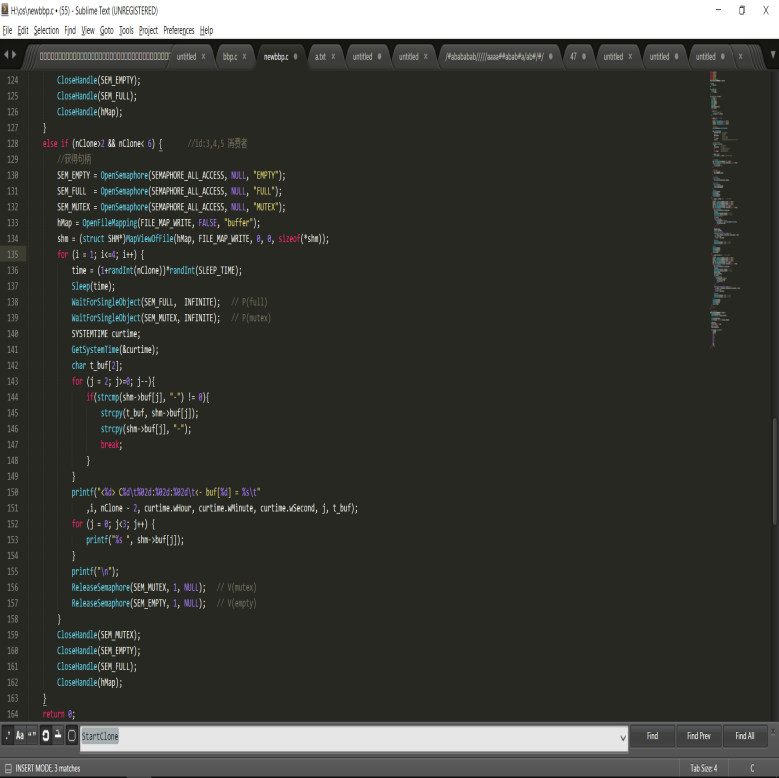
编译器：gcc 3.4.5 (mingw-vista special r3)

1. 实验步骤

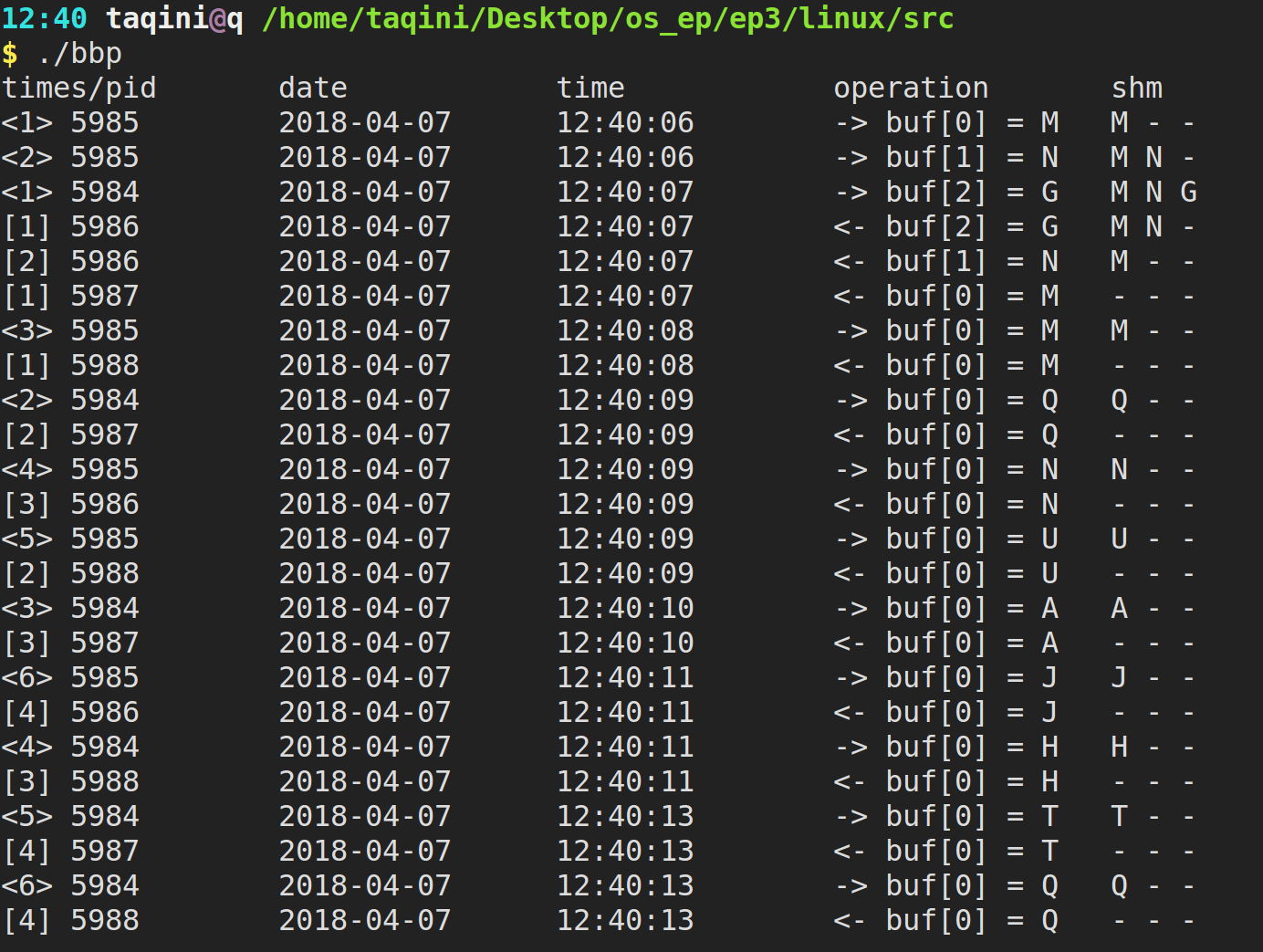
3.1 在Linux环境实现

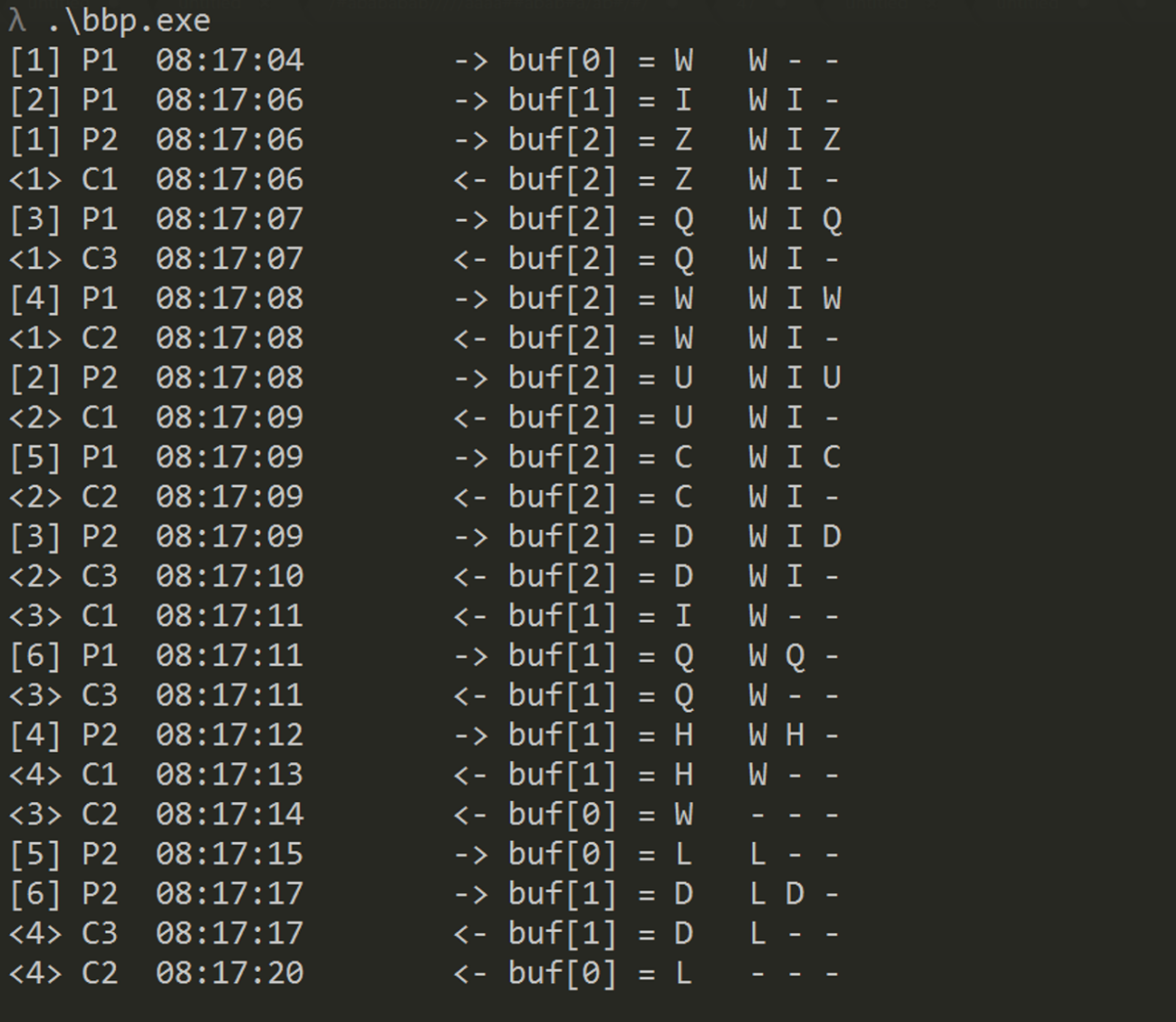
* 申请并初始化共享主存区  
  
* 申请并初始化信号量  
  其中full和empty为同步信号，分别代表已生产项目数量和缓冲区的剩余空间mutex为互斥信号，是一个二值信号灯。
* 创建5个进程，前2个作为生产者，剩余的作为消费者  
    
  　　生产者在生产数据之前，对信号量empty执行P操作，使empty的值减1，生产数据后，对信号量full执行V操作，使full的值加1。若生产者在生产数据前，缓冲区已满(empty=0)，则P(empty)操作将阻塞进程，直至empty>0后再执行生产数据的操作。通过同步信号量机制实现了“若缓冲区已满，则等待消费者取走数据后再添加”的实验要求。生产者部分代码如下：  
    
  　　消费者在消费数据之前，对信号量full执行P操作，使full的值减1，消费数据后，对信号量empty执行V操作，使empty的值加1。若消费者在要消费数据前，缓冲区中没有生产者生产的数据(full=0)，则P(full)操作将阻塞进程，直到full>0后再执行消费数据的操作。通过同步信号量机制实现了“若缓冲区为空，等待生产者添加数据后再读取”的实验要求。消费者部分代码如下：  
    
  　　生产者(消费者)在生产(消费)数据的前后，分别对互斥信号量mutex执行P操作和V操作，这保证了生产(消费)过程的原子性质，可以避免多个生产者(消费者)之间对共享资源的竞争。

3.2 在Windows环境实现

* 建立并初始化共享主存区  
  
* 建立并初始化信号量  
  
* 建立5个子进程  
    
  其中StartClone()函数中调用Win32API CreateProcess()创建子进程。nClone为进程的编号，其中0号为主进程，1-2号为生产者进程，3-5号为消费者进程。  
    
  在生产者进程中，首先获取信号量和共享主存区的句柄，用于对其进行操作，使用WaitForSingleObject()实现P操作，ReleaseSemaphore()实现V操作。其余部分与linux基本相同。生产者部分代码如下：  
    
  消费进程同理,代码如下：  
  

1. 实验结果

4.1 在Linux环境下的实验结果  


4.2 在Windows环境下的实验结果  


1. 实验总结

生产者与消费者问题的核心是“如何避免多进程对共享资源的竞争”，为了解决这个问题，引入了同步信号量来解决生产者进程和消费者进程之间对共享资源竞争的问题，引入了互斥信号量来解决多个生产者(或消费者)之间对共享资源竞争的问题。

最初我做这个实验的时候，没有使用sys/sem.h里面定义的信号量机制，而是将3个信号量放入共享主存区，在生产者或消费者生产或消费数据的前后对信号量进行加或减，若信号量的值小于0则阻塞。我希望通过这种方式让多个进程之间共享信号量的信息，但是结果还是出现因为各种竞争导致的程序结果错误。分析其原因，是因为信号量本是就是为控制共享主存区的访问而设置的，它不能作为共享主存区的一部分。  
 我做实验一(编译内核)时的源代码还留着，于是找到了信号量机制的源码/ipc/sem.c，粗略的浏览一下，大概是使用了很多种lock，才实现了对临界区的保护，于是放弃了自己实现信号量的想法...  
