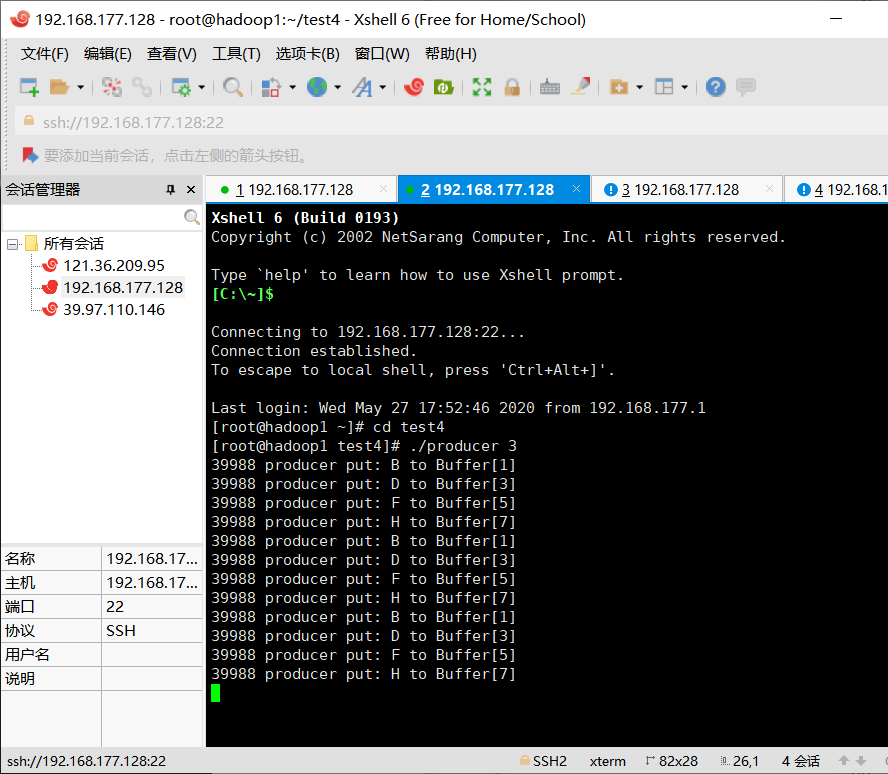
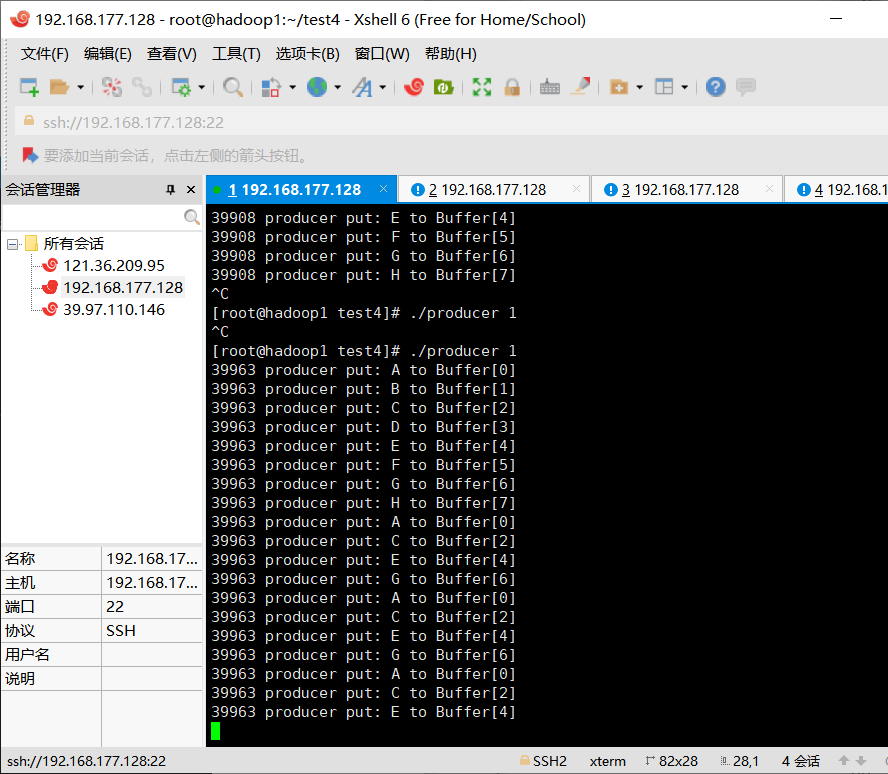
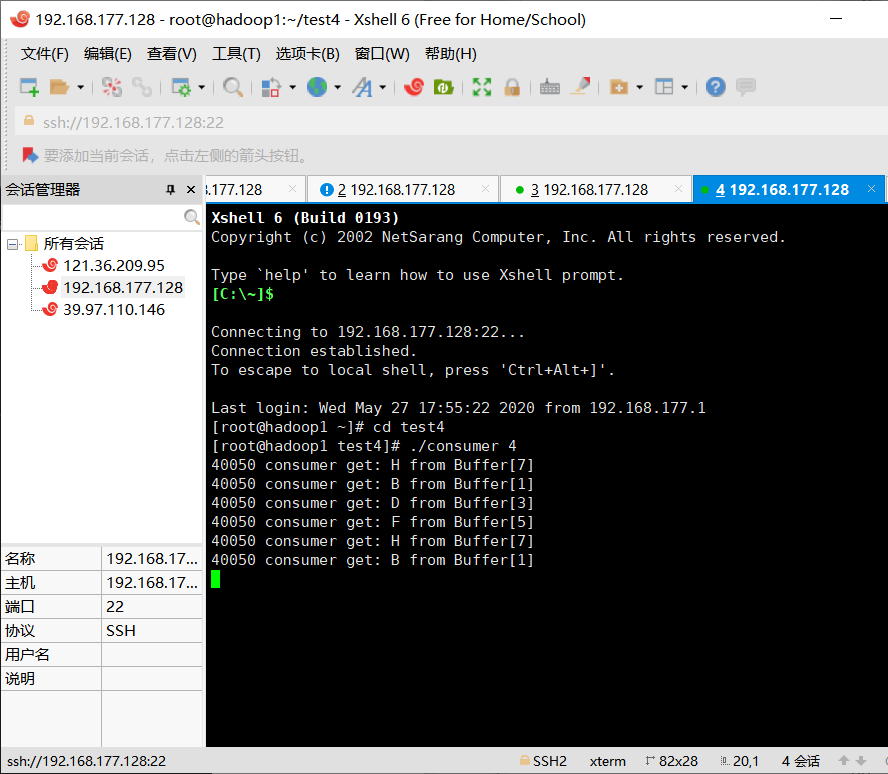
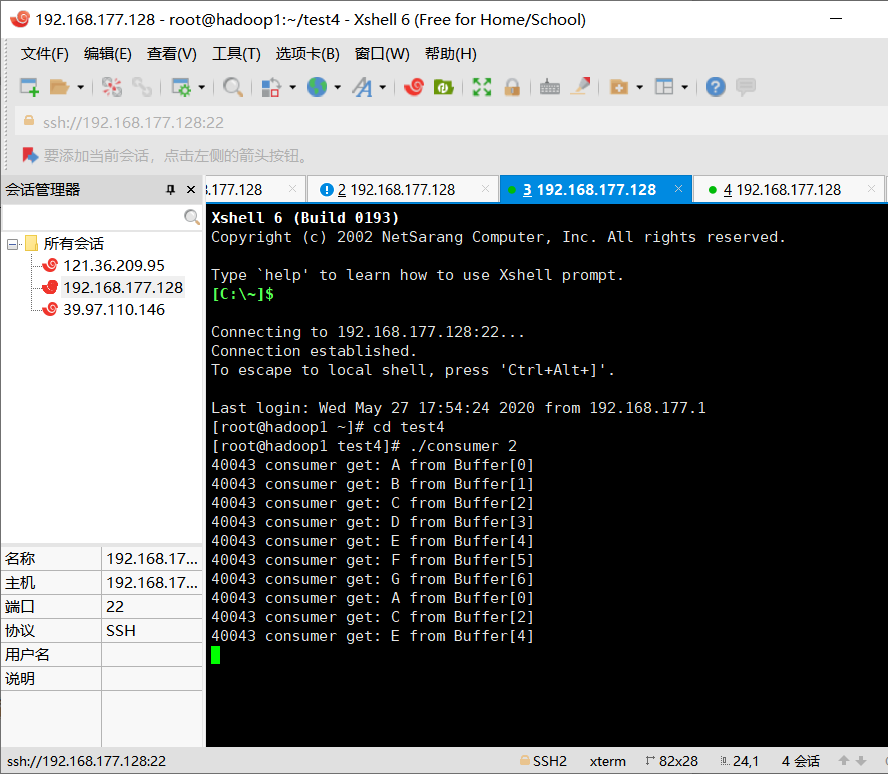
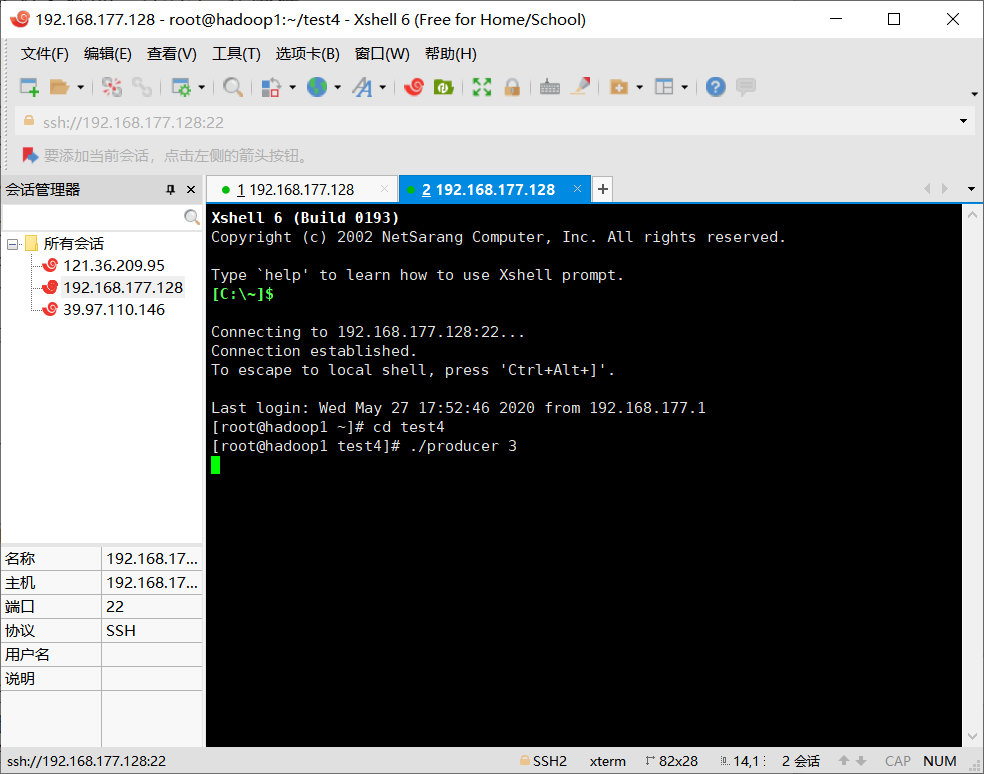
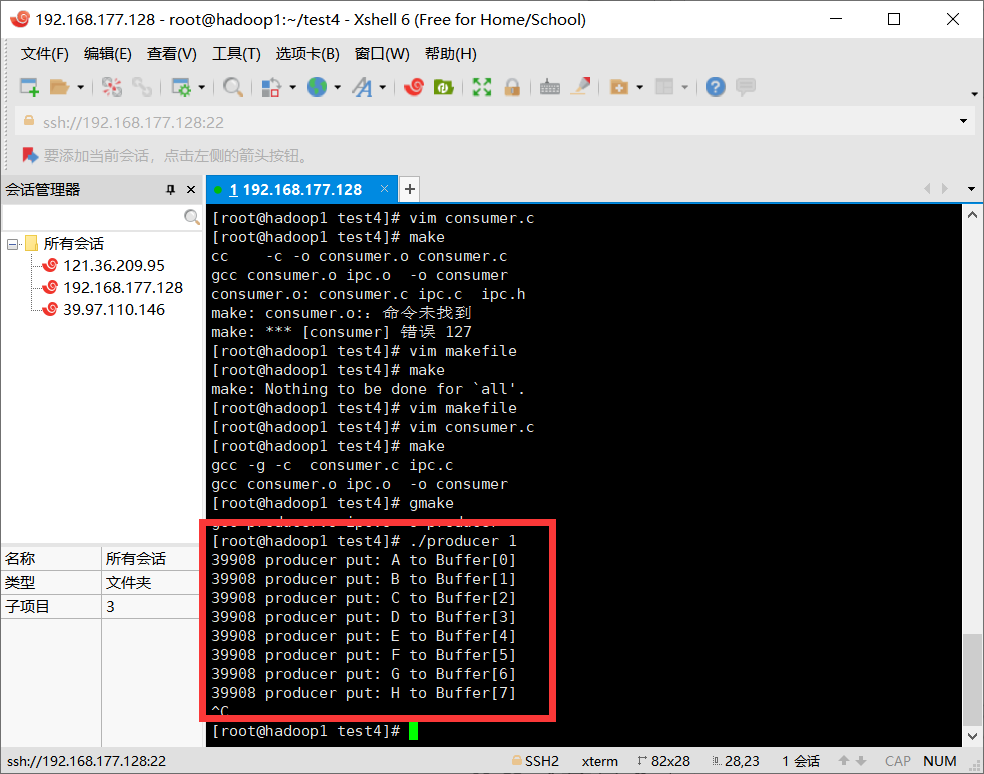
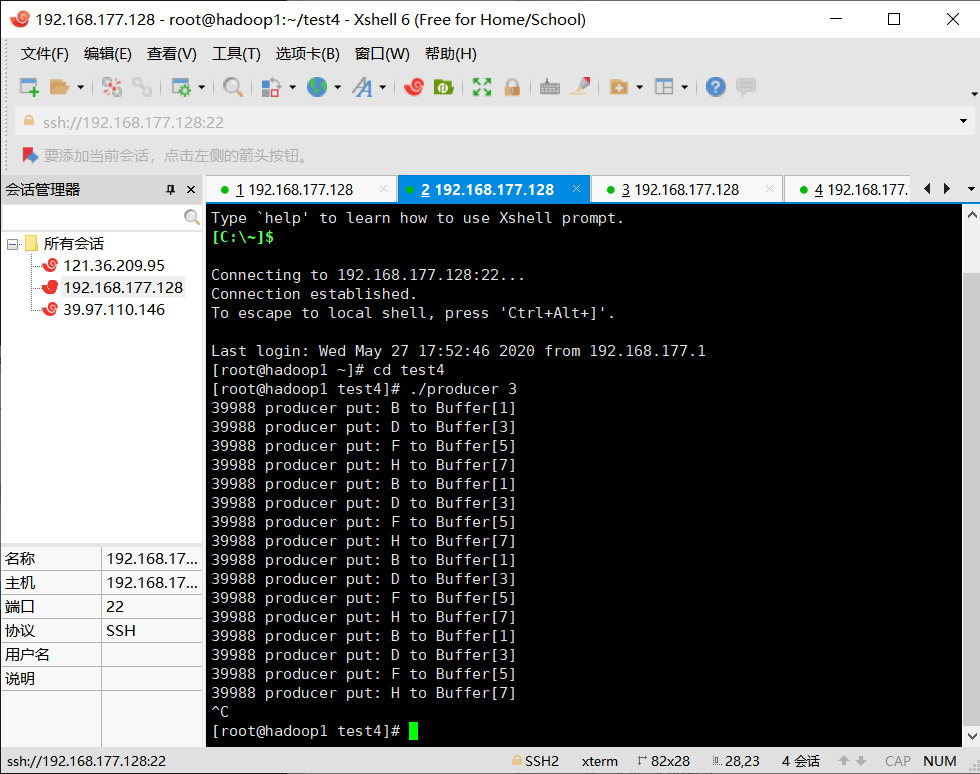
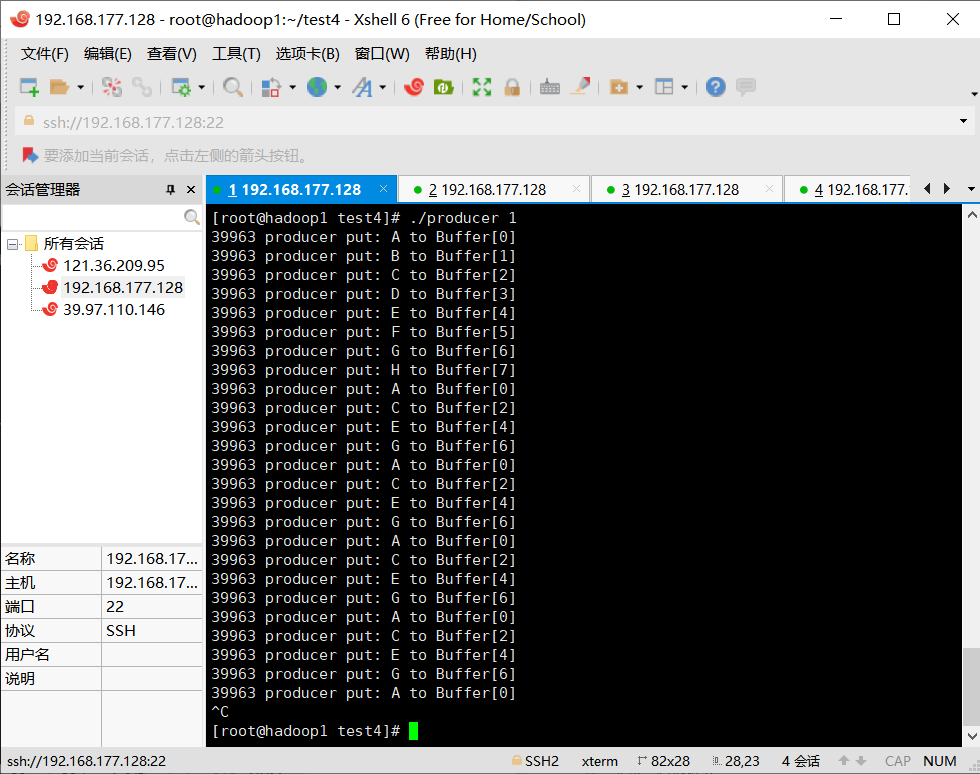
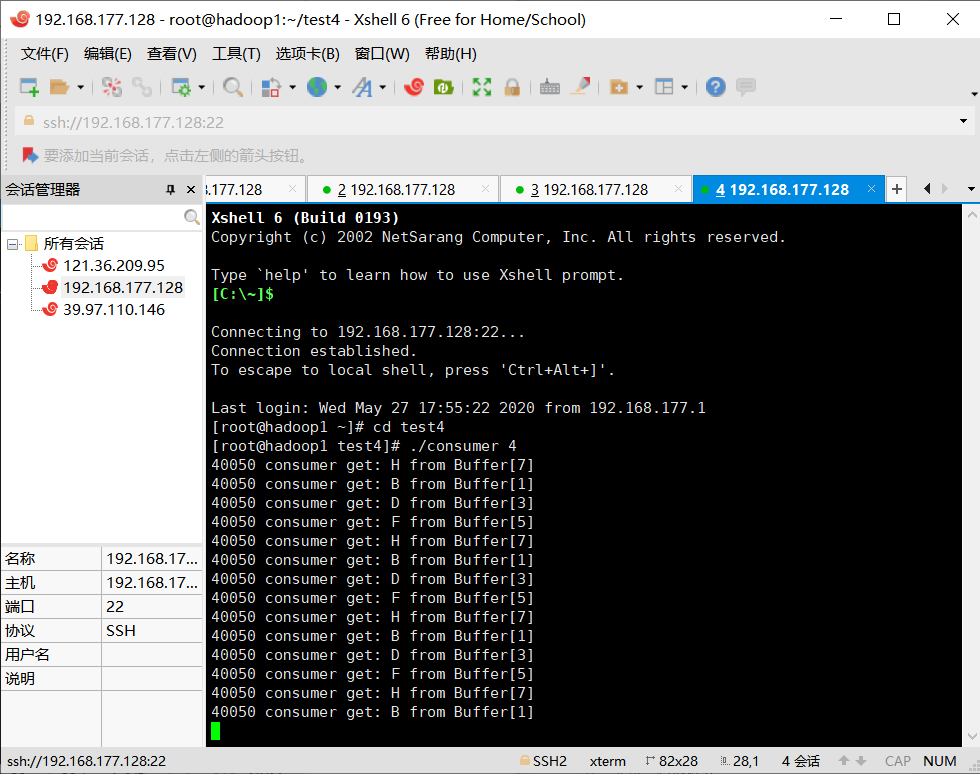
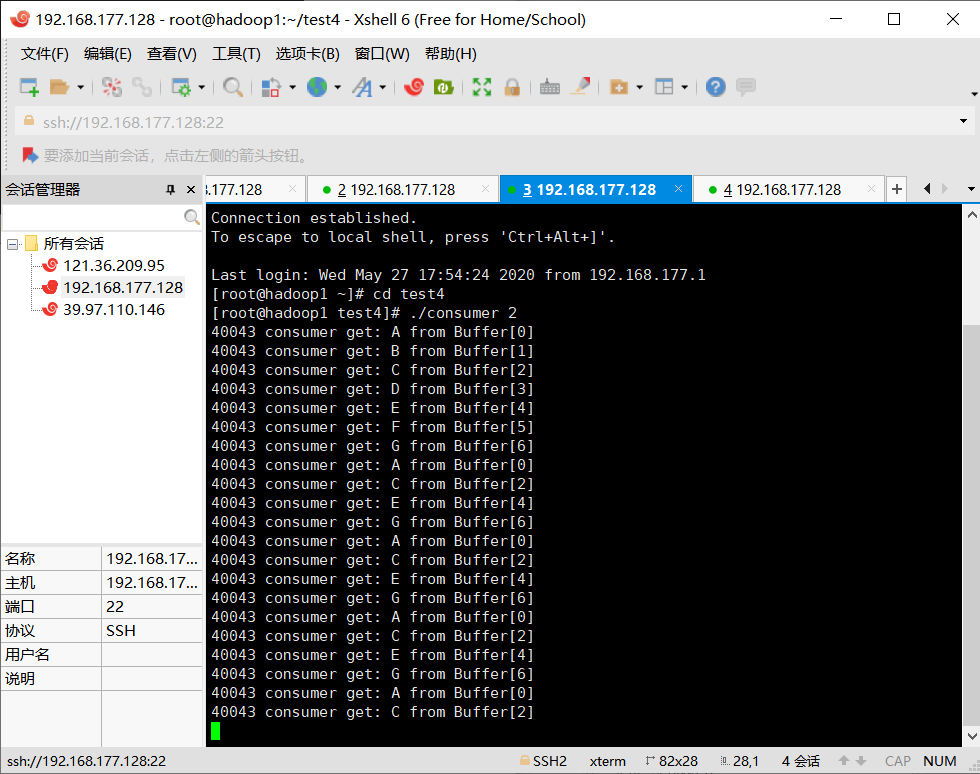
**写在前面：可以通过窗口页码来判定是第几号系统哦！由于4为上限，否则就重新计数1了，所以演示四个系统不同工作。**

1.文档中4.3示例实验运行结果。



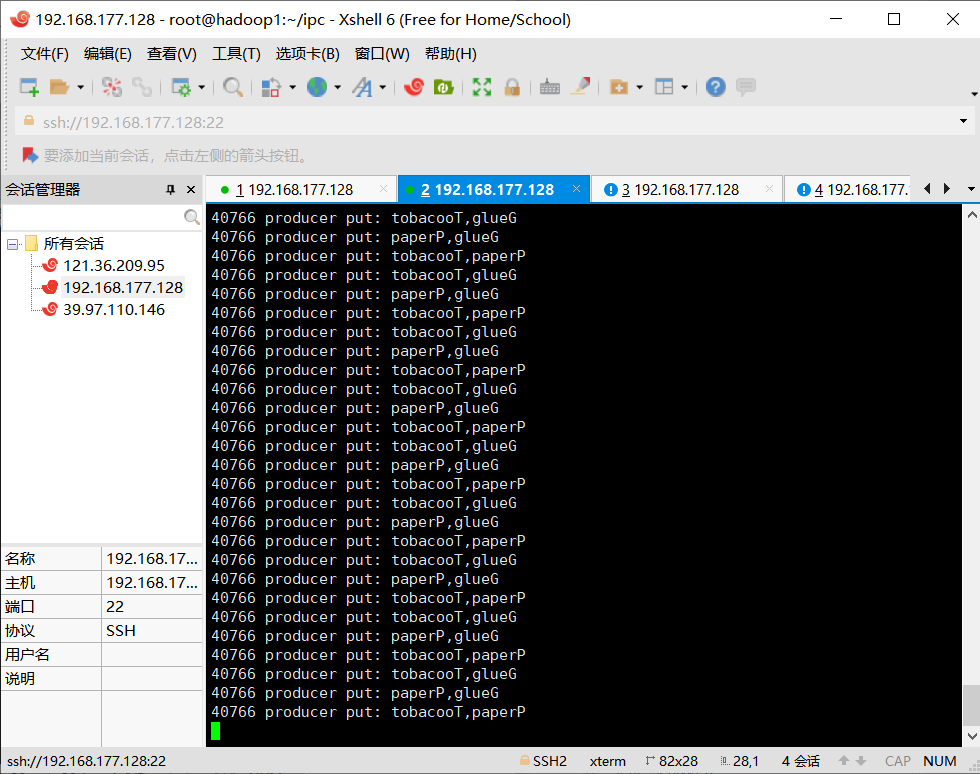
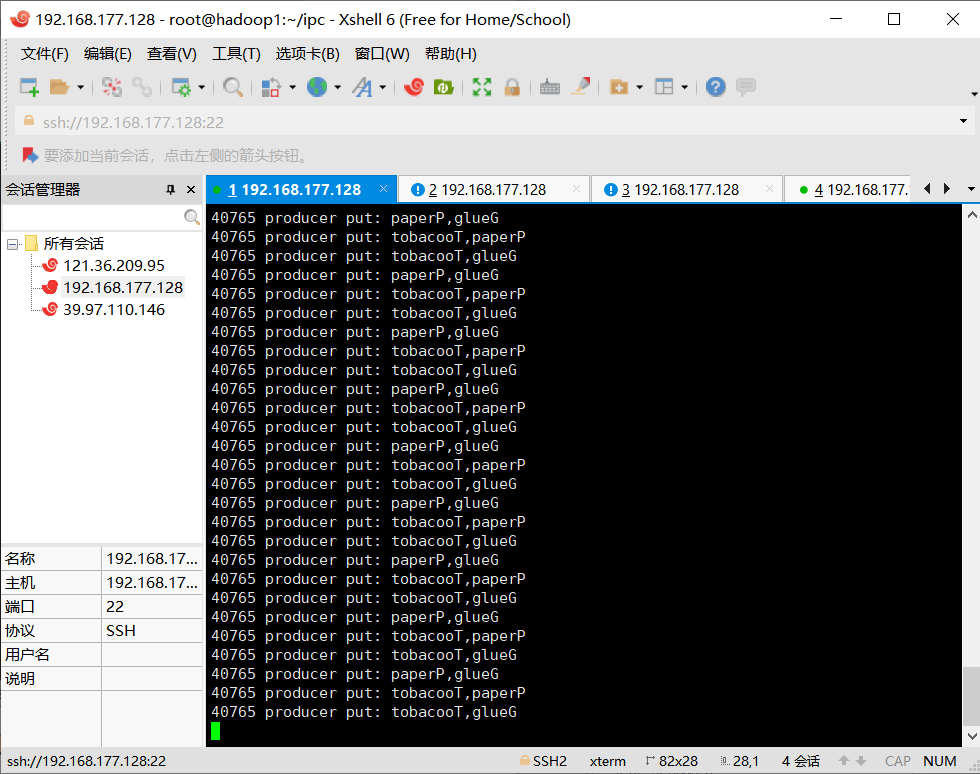
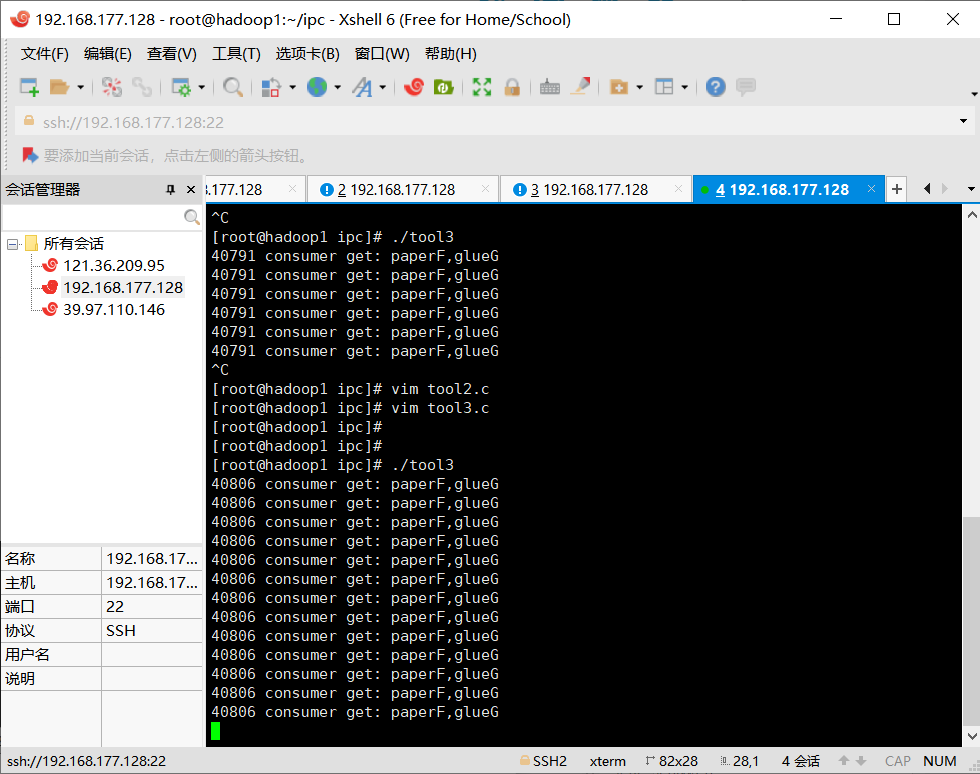
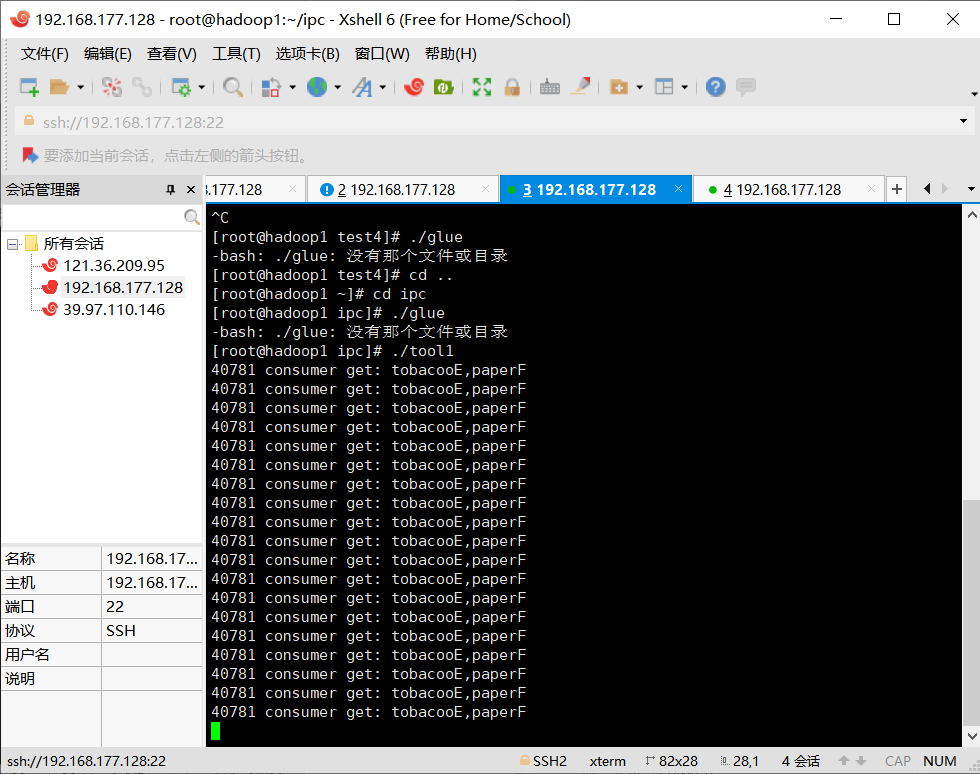
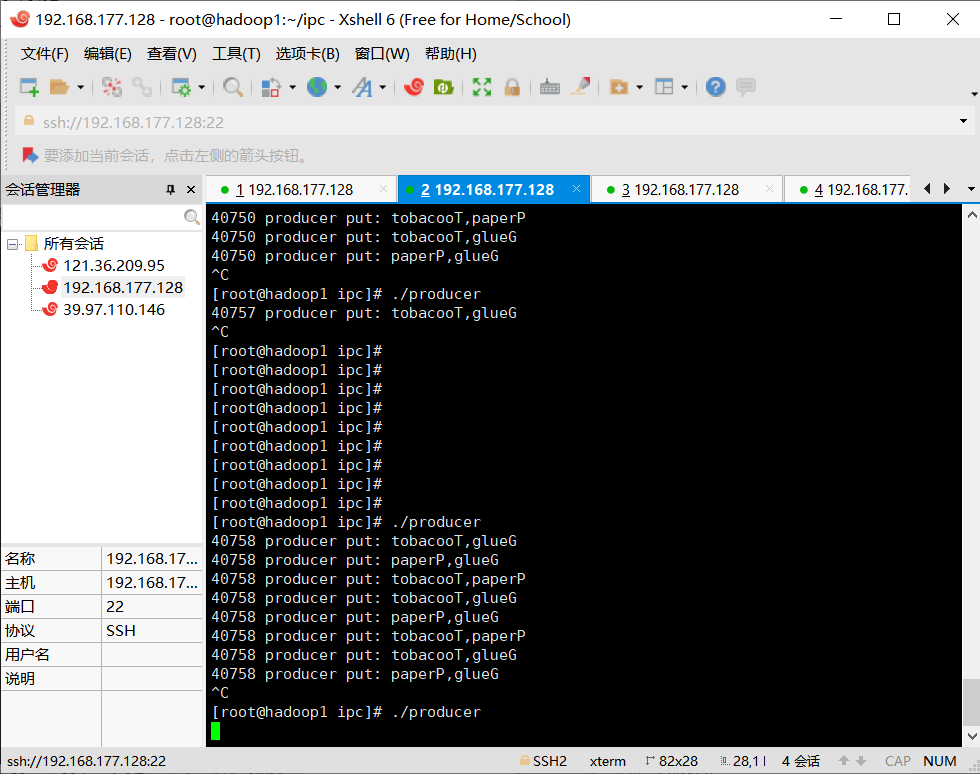
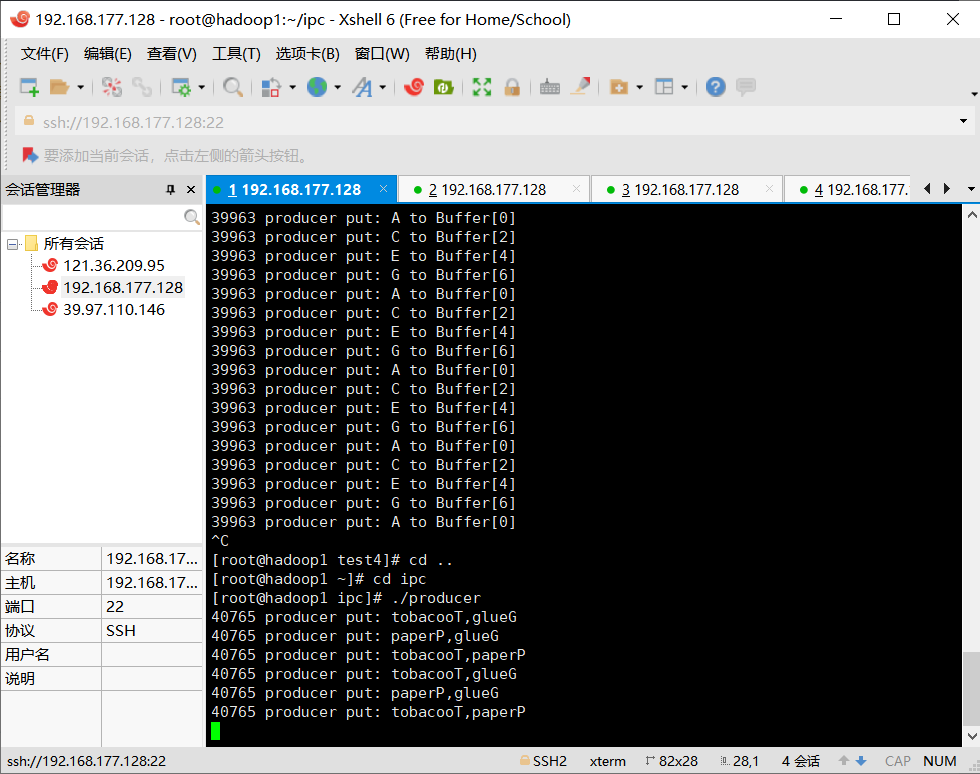
ctrl+C 将两生产者进程打断后，两消费者进程在读空缓冲区后阻塞。反之，ctrl+C 将两消费者进程打断后，两生产者进程在写满缓冲区后阻塞。



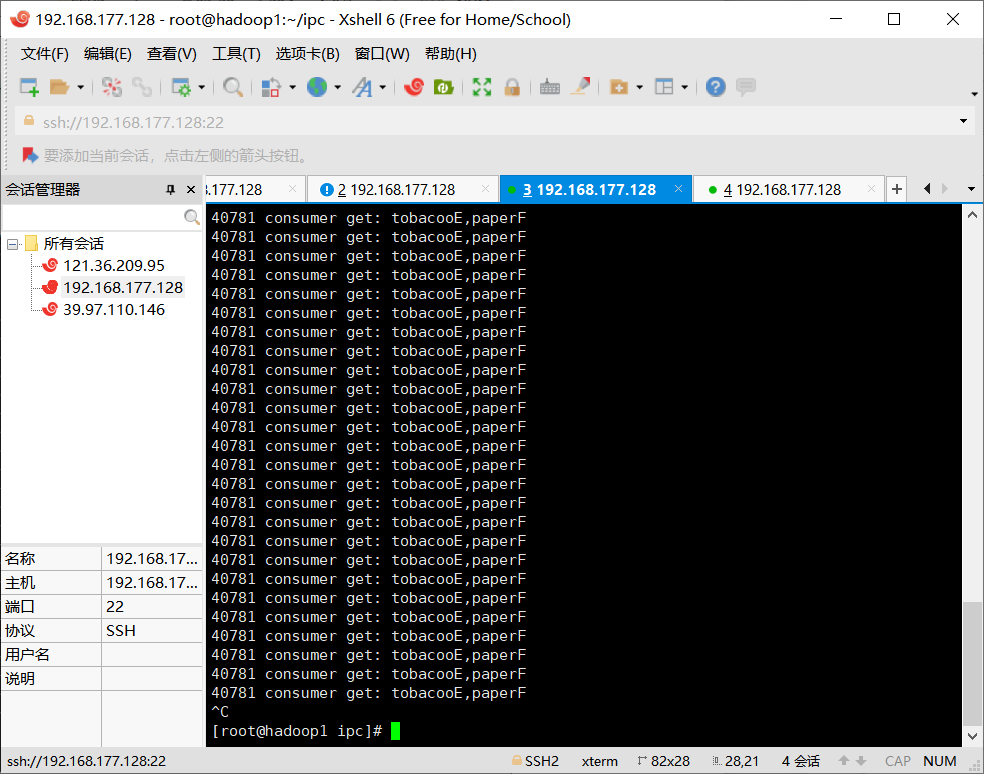
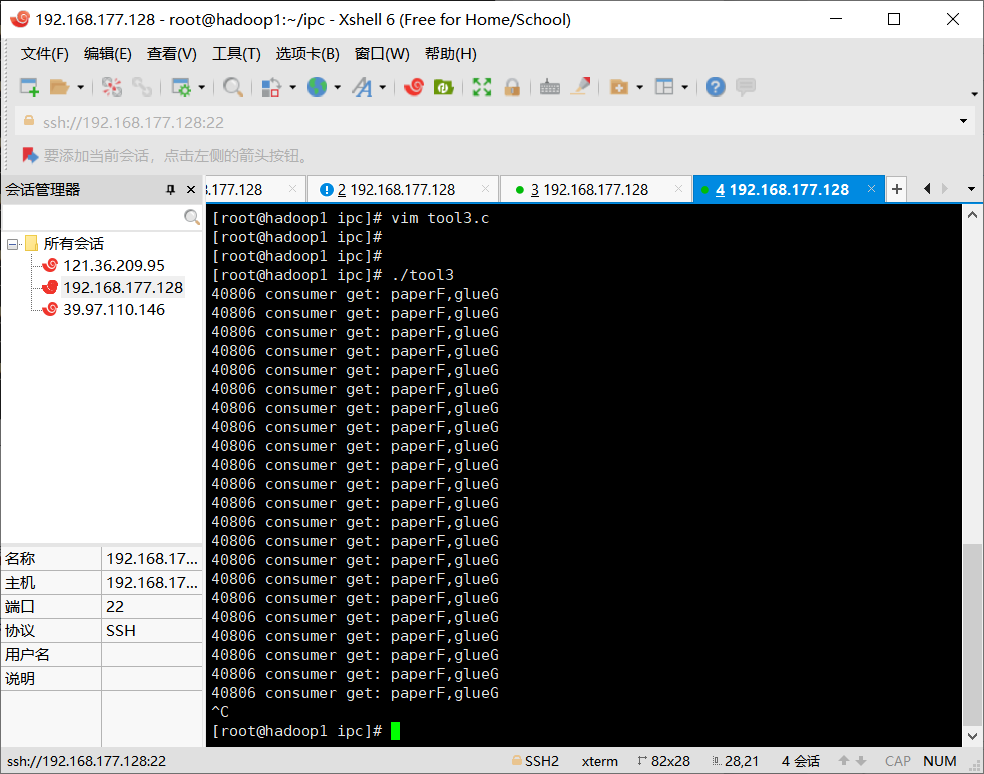


2.完成文档中4.4独立实验，回答以下内容。

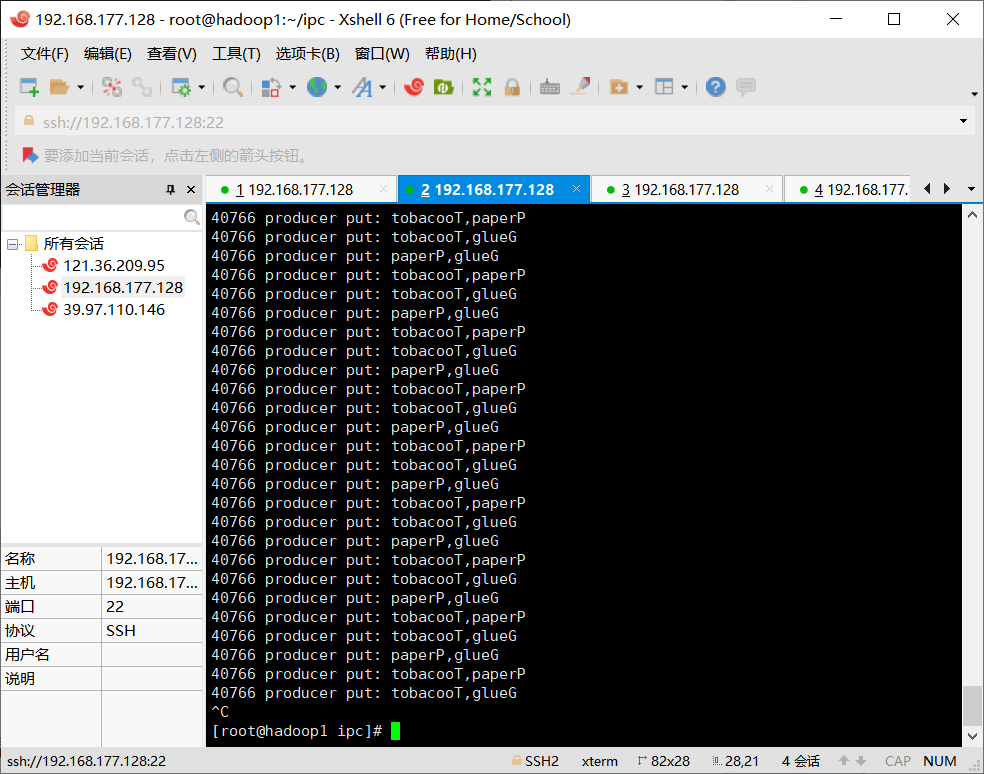
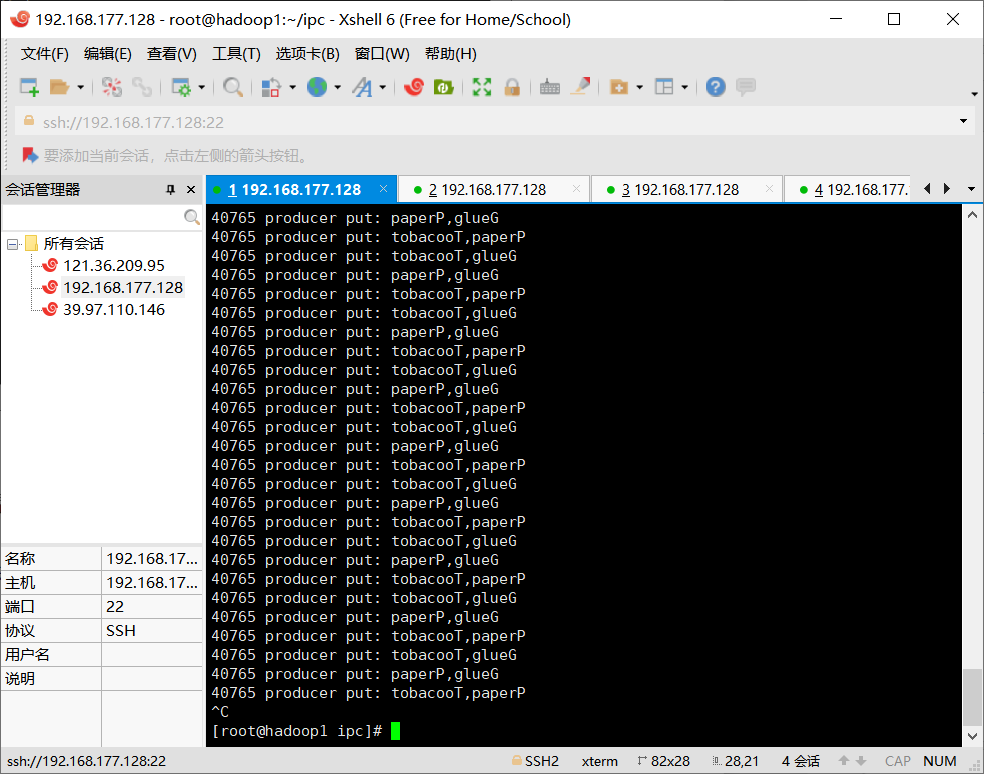
2.1 完成独立实验，使用IPC同步机制编程实现独立实验要求的功能。



上面几张截图表示生产者与消费者工作时的状态截图。



停止后1和2生产者生产满了，也停止了。（截图也看不出来。。。）



**2.2 操作系统中提供的并发进程同步机制是怎样实现和解决同步问题的？它们是怎么样应用操作系统教材中讲解的进程同步原理的？**

OS里面用的一般是管程来实现和解决同步问题。我不知道这题是不是这个答案，但可以肯定的是，信号量机制存在的问题特别明显：编写程序困难、易出错（本次上机中深有体会）。这里不再介绍管程的定义，只介绍管程的特征：

1. 局部于管程的数据只能被局部于管程的过程所访问
2. 一个进程只有通过调用管程内的过程才能进入管程访问共享数据
3. 每次仅允许一个进程在管程内执行某个内部过程

要注意：每次仅允许一个进程在管程内执行某个内部过程。这就造就了同步功能。

我们可在管程中设置条件变量及等待/唤醒操作以解决同步问题。可以让一个进程或线程在条件变量上等待(此时、该进程应先释放管程的使用权,也就是让出“入口”);可以通过唤醒操作将等待在条件变量上的进程或线程唤醒。程序员可以用某种特殊的语法定义一个管程,之后其他程序员就可以使用这个管程提供的特定“入口”很方便地使用实现进程同步了。

**2.3 信号灯机制是怎么样完成进程的互斥和同步的？信号量的初值和其值的变化的物理意义是什么？（第二个问题我先回答了，助教注意一下）**

注：以下代码中的P代表加锁操作，即让其它进程知晓该资源正在被其他进程使用；V代表解锁操作，即让其他进程知晓该资源没有被其他进程占有。我们可以理解，PV操作在某种程度上其实就是在改变信号量的值，进程只需在特定语句中加上if/while判断该值是否等于1或者0来简单判断该资源是否被占有（在互斥中是这种用法）或者说进程实现先后关系（在同步中是这种用法）。初值一般是平常状态，即未被占有或者允许进程执行状态。

互斥：

1.分析并发进程的关键活动,划定临界区(如:对临界资源打印机的访问就应放在临界区)

2.设置互斥信号量 mutex,初值为1

3.在临界区之前执行P(mutex)

4.在临界区之后执行V(mutex)

同步：

1.分析什么地方需要实现“同步关系”,即必须保证“一前一后”执行的两个操作(或两句代码）

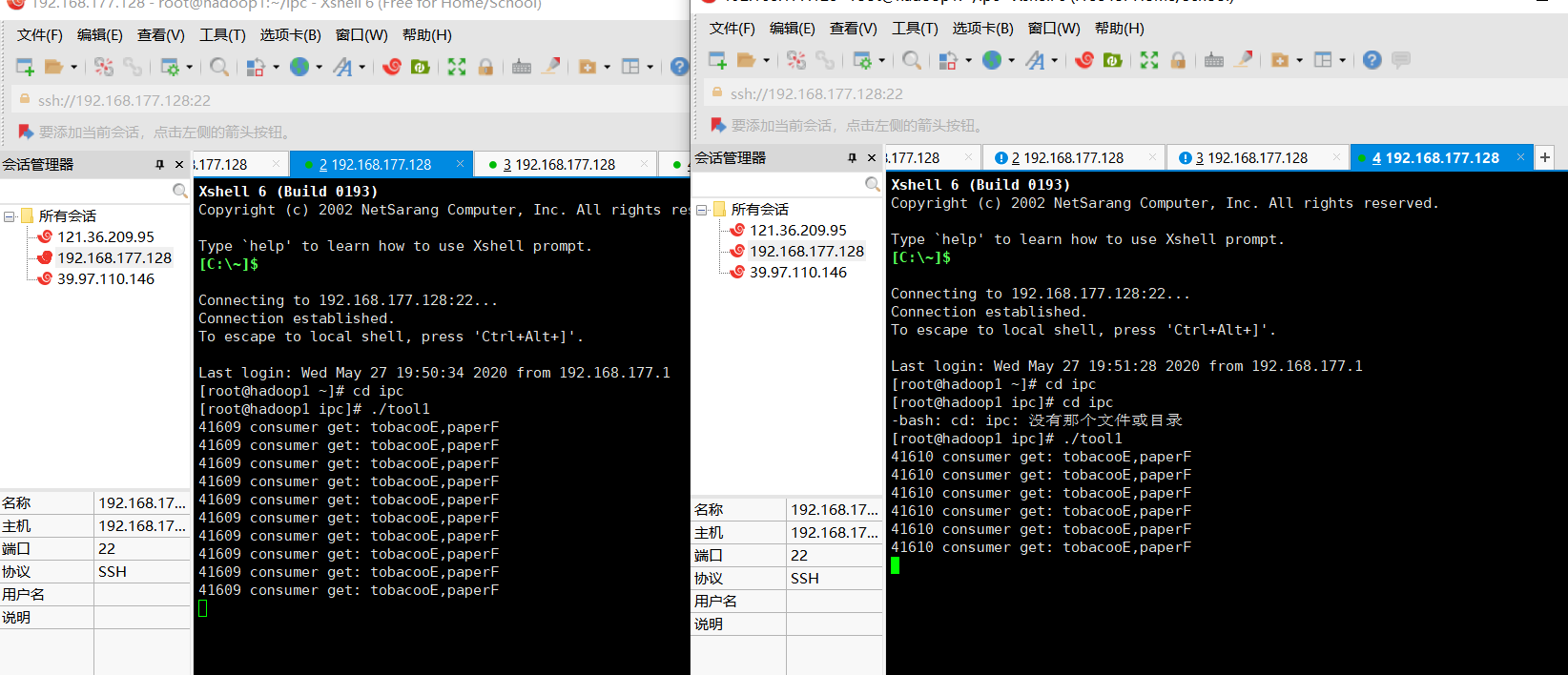
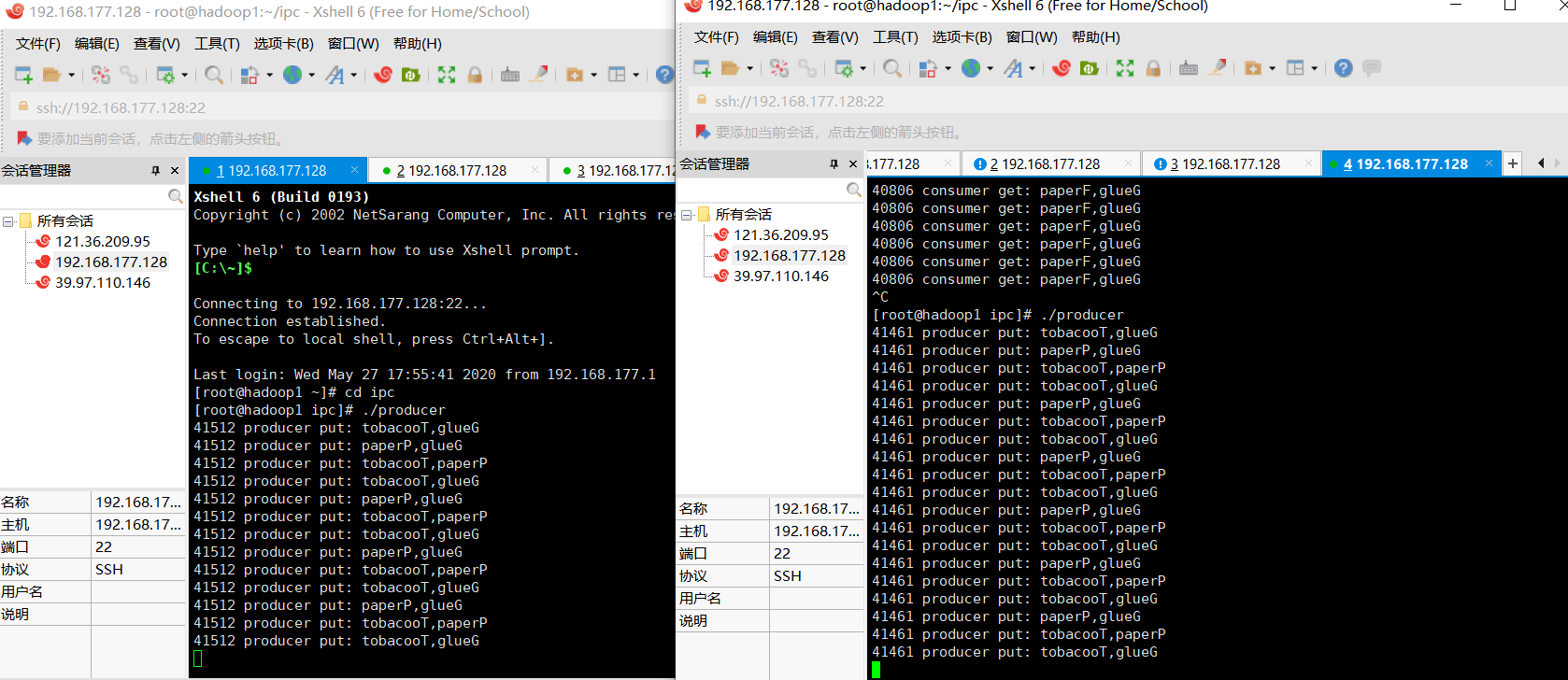
2.设置同步信号量S,初始为0

3.在“前操作”之后执行V(S)

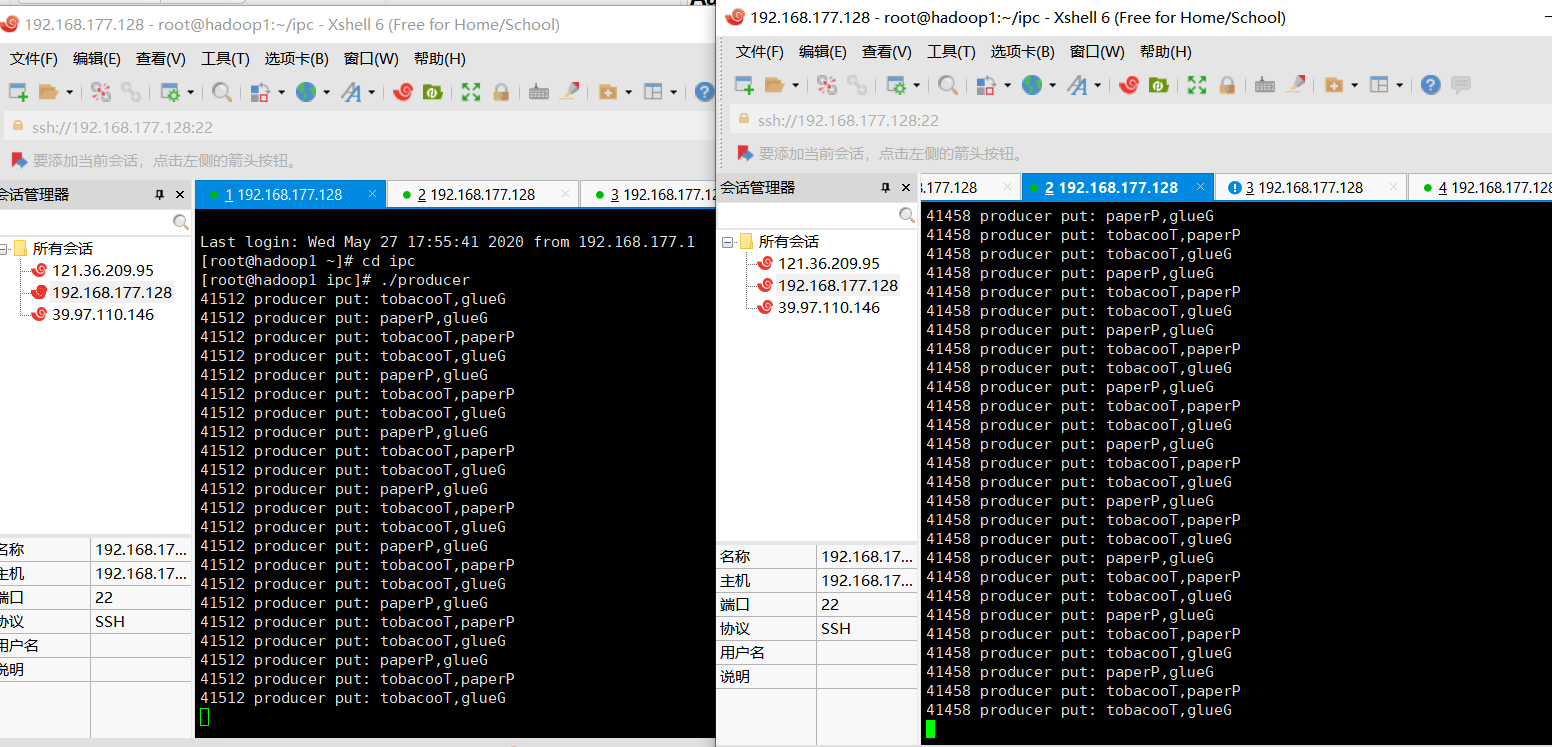
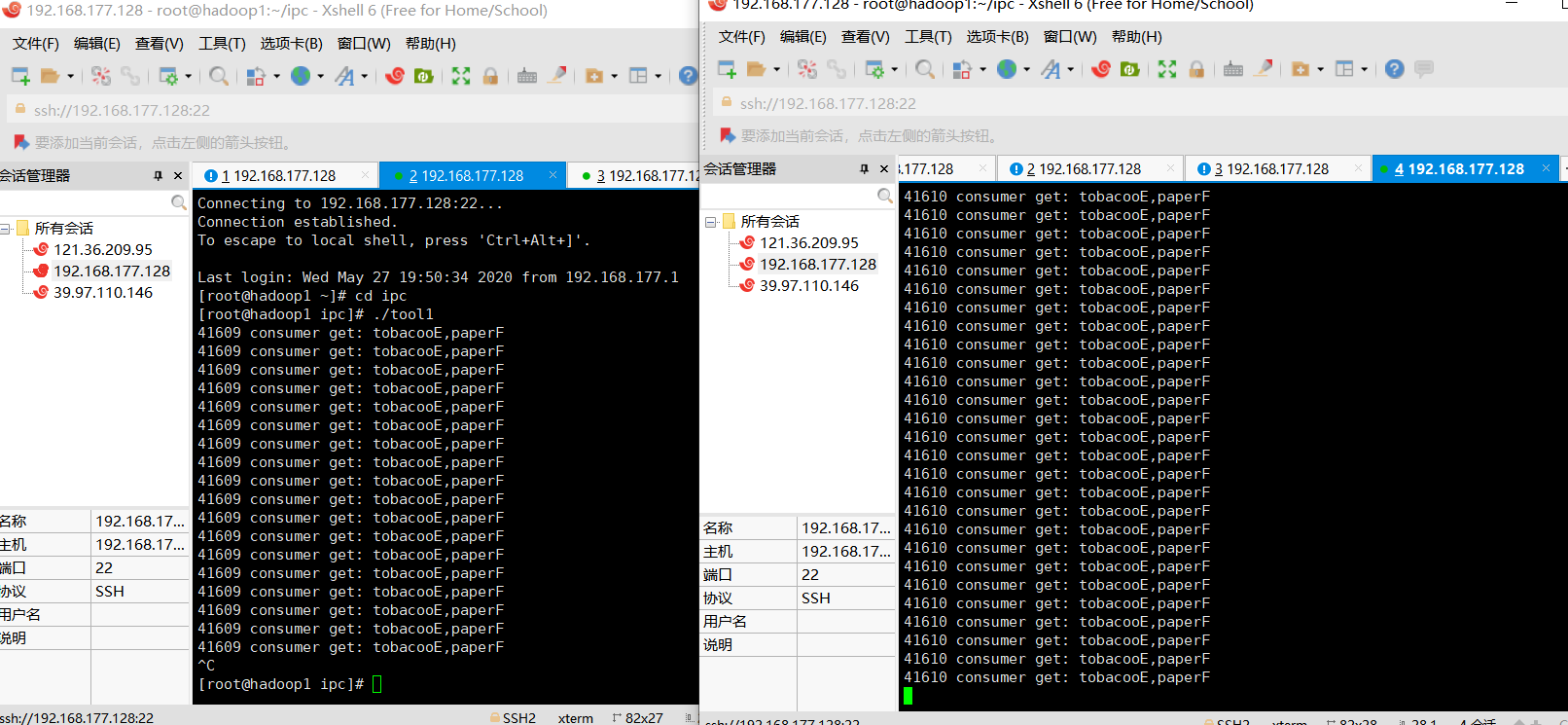
4.在“后操作”之前执行P(S)

**2.4 使用5个生产者和消费者，以不同的启动顺序、不同的执行速率检测独立实验程序是否能够满足同步的要求。**

可以的。不同启动顺序可以做到，不同执行速率是通过不同时间段停止该进程来实现的，最终消费者和生产者都会有相关反应。



停止一方生产者后，消费者会有些许卡顿。（可以理解成栈已空）



停止一方消费后，生产者会有些许卡顿。（可以理解成栈已满）

（这里是开了十个进程的，但是这个截图有点问题。。如果截到生产者，消费者就截不清楚了）