**实 验 报 告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **组 别** |  | **姓 名** | 尹浩男 | **同组实验者** |  |
| **实验项目名称** | 进程同步：生产者\消费者问题 | | | **实验日期** | 11月16,23日 |
| **教师评语:** |  | | | | |
| **实验成绩：** | | | **指导教师（签名）：**  2020年 月 日 | | |
| **一．实验目的**  1. 理解POSIX线程机制  2. 理解POSIX多线程的创建方法  3. 掌握POSIX多线程的同步机制  **二．实验内容**  在系统中有个缓冲区（每个可以放1个产品）构成的仓库。有个生产者，每个生产者可以生产个产品。有个消费者，每个消费者消费若干个产品，直到消费完所有的产品为止，即，其中是消费者实际消费的产品个数。  **输入**：生产者个数、消费者个数、缓冲区个数、每个生产者生产产品的个数等。  **输出**：生产者-消费者并发执行的过程、每个进程的状态变化。如：  （1）生产者将第X个产品放入仓库；  （2）消费者从仓库中消费第Y个产品；  （3）生产者阻塞；  （4）消费者阻塞；  （5）生产者被唤醒；  （6）消费者被唤醒。  **三. 实验步骤**  **编译**  gcc producerConsumer.c -o producerConsumer -lpthread  执行  ./producerConsumer  2 3 2  1 2  3 3 2  **四. 实验结果**  **输入的三个数是 生产者n，消费者m，缓冲区大小b**  实验结果如下图所示：    **五. 分析**  上述程序需要注意的是虚假唤醒的问题，当线程被唤醒，但是没有足够的资源让它去执行事件，虚假唤醒会导致 缺少产品的时候还能进行消费，  例如标准的模板是   |  | | --- | | pthread \_mutex\_lock(&mutex)  while或if(线程执行的条件是否成立) // 当有多个线程可能同时在等待一个锁的时候 需要使用while而不是if  pthread\_cond\_wait(&cond, &mutex);  线程执行  pthread\_mutex\_unlock(&mutex); | | 需要解释的有两点，为什么要加锁，以及为什么可以使用while和if。首先解释第一点，有两个方面，线程在执行的部分访问的是进程的资源，有可能有多个线程需要访问它，为了避免由于线程并发执行所引起的资源竞争，所以要让每个线程互斥的访问公有资源，但是细心一下就会发现，如果while或者if判断的时候，不满足线程的执行条件，那么线程便会调用pthread\_cond\_wait阻塞自己，但是它持有的锁怎么办呢，如果他不归还操作系统，那么其他线程将会无法访问公有资源。这就要追究一下pthread\_cond\_wait的内部实现机制，当\*\*pthread\_cond\_wait被调用线程阻塞\*\*的时候，\*\*pthread\_cond\_wait会自动释放互斥锁\*\*。释放互斥锁的\*\*时机\*\*是什么呢：是线程从调用pthread\_cond\_wait到操作系统把他\*\*放在线程等待队列之后\*\*，这样做有一个很重要的原因，就是mutex的第二个作用，保护条件。想一想，线程是并发执行的，如果在没有把被阻塞的线程A放在等待队列之前，就释放了互斥锁，这就意味着其他线程比如线程B可以获得互斥锁去访问公有资源，这时候线程A所等待的条件改变了，但是它没有被放在等待队列上，\*\*导致A忽略了等待条件被满足的信号\*\*。倘若在线程A调用pthread\_cond\_wait开始，到把A放在等待队列的过程中，都持有互斥锁，其他线程无法得到互斥锁，就不能改变公有资源。这就保证了线程A被放在等待队列上之后才会有公有资源被改变的信号传递给等待队列。 |   pthread是POSIX提供的重要线程工具包，pthread创建的线程对Linux来说是核外线程，也就是通常所说的用户线程，其调度是由pthread管理包完成的 | | | | | |