|  |
| --- |
|  |
| 使用Windows API实现生产者消费者问题  使用Windows API实现哲学家就餐问题。 |
| 1. 实验环境（列出本次实验所使用的软件、工具） |
| 1. windows主机操作系统（处理器：Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz）  2. VMwareWorkstation Pro  3. ubuntu虚拟机操作系统（64位系统；版本号：20.04.1） |
| 1. 实验思路及过程 |
| （一）使用Windows线程API实现生产者消费者问题   1. 实验思路   伪代码思路如下图所示    其中信号量设置    （1）信号量的实现  互斥量hMutex保护对缓冲区插入和删除的操作，采用互斥锁实现  计数信号量hSemaphore\_empty（记录有多少空的缓冲区）、hSemaphore\_full（记录有多少满的缓冲区）    其中，对于三个信号量：  创建：    关闭：    对于P(full)和V(full)的实现：  ReleaseSemaphore(hSemaphore\_full, -1, NULL)实现减量操作  WaitForSingleObject(hSemaphore\_full, INFINITE)实现信号量不符合条件无限等待，直至可以实现      对于P(hMutex)和V(hMutex)的实现：      （2）线程的创建与同步  通过CreateThread()函数创建生产者消费者线程    调用produce函数实现，传入produce\_num[1]等参数表示生产者/消费者的编号  通过 WaitForMultipleObjects()函数结束全部线程：    （3）缓冲区的实现  缓冲区包括一个固定大小的数组，元素类型为buffer\_item，数组可以按照循环队列处理。  创建与初始化：    通过insert\_item()和remove\_item()函数来完成对缓冲区的操作，分别用于生产者和消费者线程。  insert\_item()    remove\_item()    （4）生产者与消费者线程的框架  生产者线程不断交替执行如下两个动作：睡眠一段时间，向缓冲区插入一个随机数。随机数由函数rand（）生成，值位于0和RAND\_MAX之间。消费者也睡眠一段随机时间，当唤醒时会从缓冲区内去除一项。  消费者线程：    生产者线程：  2.实验中遇到的问题：   1. 伪代码实现错误   因为第一次接触Windows API的编程，对其实现与部分函数了解不清晰，导致P()操作和V()操作实现出现偏差，每条线程进行一次就已结束不再循环进行。   1. 信号量和互斥量的初值设置问题   开始因为粗枝大叶，将empty的初值和缓冲区的循环队列长度设置不一致，导致缓冲区未满就被锁定。  （三）使用Windows API实现哲学家就餐问题  1.实现思路  伪代码实现：  其中采取n位哲学家就餐时，最多允许n-1位哲学家同时进餐来避免死锁现象。    信号量设置    2.实现步骤  部分信号量实现不再赘述。  进餐与思考通过rand函数与Sleep函数实现随机时间的等待。      phi\_mutex与chopstick[PHI\_num]信号量设置 |
|  |