临界资源Critical Resource

Producer-Consumer

Dijkstra|信号量机制|整型信号量

利用整型信号量来模拟解决生产者-消费者问题

**问题的描述**

有一群生产者进程在生产产品，并将这些产品提供给消费者进程去消费。为使生产者进程与消费者进程能并发执行，在两者之间设置了具有些许缓冲区的缓冲池，生产者进程将他所生产的产品放入一个缓冲区；消费者进程可从一个缓冲区中取走产品去消费。尽管所有的生产者进程和消费者进程都是以异步方式运行的，但他们之间必须保持同步，即不容许消费者进程到一个空缓冲区去取产品，也不容许生产者进程向一个已装满产品且尚未被取走的缓冲区中投放产品。

**数据模拟**

利用一个数组来表示上述的具有n个(0,1,2,…,n-1)缓冲区的缓冲池。使用输入指针in来指示下一个可投放产品的缓冲区，每当生产者进程生产并投放一个产品后，输入指针加1；使用输出指针out来指示下一个可从中获取产品的缓冲区。没当消费者进程消费一个产品后，输出指针加1。这里的缓冲池是组织成循环缓冲的，故应把输入指针加1表示成：in = (in+1) % n；输出指针加1表示成：out = (out + 1) % n。

其中，当(in+1)%n==out表示缓冲池已满；而in==out表示缓冲池为空。

此外，引用一个整型变量counter，其初始值为0。每当放入一个产品后，counter++；每当消费一个产品之后，--counter。

**整型信号量解决同步冲突**

由于在Windows系统下无法直接创建进程，此处利用线程来模拟。

整型信号量的定义：一个用于表示资源数目的整型量S，它与一般的整型量不同，除初始化外，仅能通过两个标准的原子操作wait()与signal()来访问。

**模拟设计**

为方便模拟描述，程序中共创建了3个线程，一个是主线程main，另外两个分别是producer线程和consumer线程。线程main()主要实现producer线程和consumer线程的创建操作，并无其他作用。producer线程则不断给临界资源（缓冲池）添加产品，consumer线程则不断在临界资源（缓冲池）取走产品。

采用的平台：win32+visual studio开发工具

涉及到的核心win32函数：

CreateThread():创建线程，使用该函数创建producer线程和consumer线程。

CloseHandle():释放创建线程时产生的句柄。

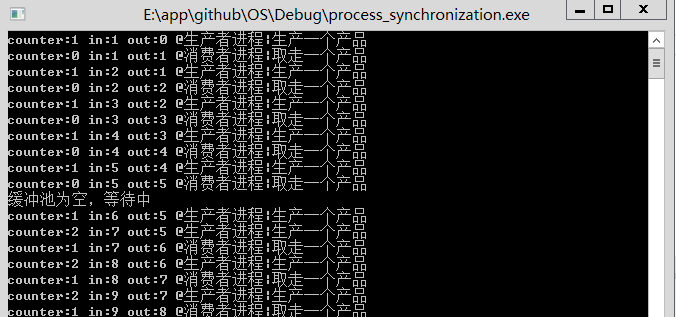
Sleep():执行挂起一段时间；注意在win32环境下，Sleep()函数的第一个字母必须大写。

CreateMutex():创造一个独占资源；

WaitForSingleObject():申请资源；

ReleaseMutex():释放一个独占资源。

其中, CreateMutex()、WaitForSingleObject()、ReleaseMutex()函数的使用，主要是为了实现wait()和signal()操作的原子性，即它们的执行是不可中断的。程序执行效果截图如下：



**核心代码：**