操作系统

实验报告

实验名称：进程同步

院系：计算机学院·网络空间安全学院

班级：

学号：

姓名：

指导老师：匡林爱

日期：2020年10月11日

1. **实验目的和要求：**

编写一个程序，实现生产者-消费者问题(P-C问题)的同步算法。通过程序运行时的输出信息，可以清楚地看到在“生产者-消费者”进程(线程)模型中，各进程(线程)的活动情况，从而加深对进程同步问题的理解。本上机实验可使学生了解(掌握)利用Win32 API编写程序实现同步算法的一些方法。

由于同一进程的多个线程之间的通信问题实现比较简单，故本实验采用由主线程创建多个生产者线程和多个消费者线程，然后在这些线程之间实现同步，这使得本实验的程序设计相对比较简单。因此本实验中的进程同步问题实际上是线程同步问题。

1. **实验内容：**

定义类sy\_pc的一个全局对象s\_pc，其循环缓冲中缓冲区个数为10。定义全局对象的目的是简化各生产者进程(线程)和消费者进程(线程)共享信号量、缓冲区等资源，简化程序。

在类sy\_pc的构造函数中，利用Win32 API函数CreateSemaphore( )创建3个信号量对象，这3个对象的句柄分别是empty、full和mutex。empty所指的信号量对象的初值为10(即循环缓冲的缓冲区个数)；full所指的信号量对象的初值为0，即初始状态，满缓冲区个数为0；mutex所指的信号量对象的初值为1，用于互斥访问共享变量in或out。关于Semaphore对象以及相关API函数将在下面介绍。

在成员函数putbuffer( )和getbuffer( )中，利用跟Semaphore Objects有关的wait函数和release函数来实现同步。这些函数将在后面具体介绍。

主函数(主线程)中循环创建5个生产者线程和3个消费者线程，然后主线程查询各个子线程是否结束。待各个子线程结束后主线程也结束。

每个生产者线程执行ProducerThread(LPVOID p)的代码，每个消费者线程执行Consumer- Thread(LPVOID p)的代码。为演示方便，每个生产者存放9个“产品”后便结束运行，并返回数值1，以便主线程查询；每个消费者取完15个“产品”后也结束运行，返回1。程序中，5个生产者总共存放了45个产品，而3个消费者总共取了45个产品，这样规定是为了所有生产者和消费者都能运行结束。

程序中各线程对于屏幕访问的互斥，采用设置CRITICAL\_SECTION 对象cs\_Screen来实现，这在上机实验三中已经使用过。当然也可以用Mutex对象或用设置初值为1的Semaphore对象来实现对屏幕的互斥访问，就像本程序中的mutex那样。CRITICAL\_SECTION 对象只用于互斥，不能用于同步，且只能用于同一进程的各线程之间的互斥。

**相关Win32 Api函数介绍：**

**1．CreateSemaphore( ) //** 创建一个有名字的或无名字的semaphore对象

**2．WaitForSingleObject( ) //** 等待一个事件信号直至信号出现或者超时

**3．ReleaseSemaphore( ) //** 使指定的semaphore对象的count值增加给定的值

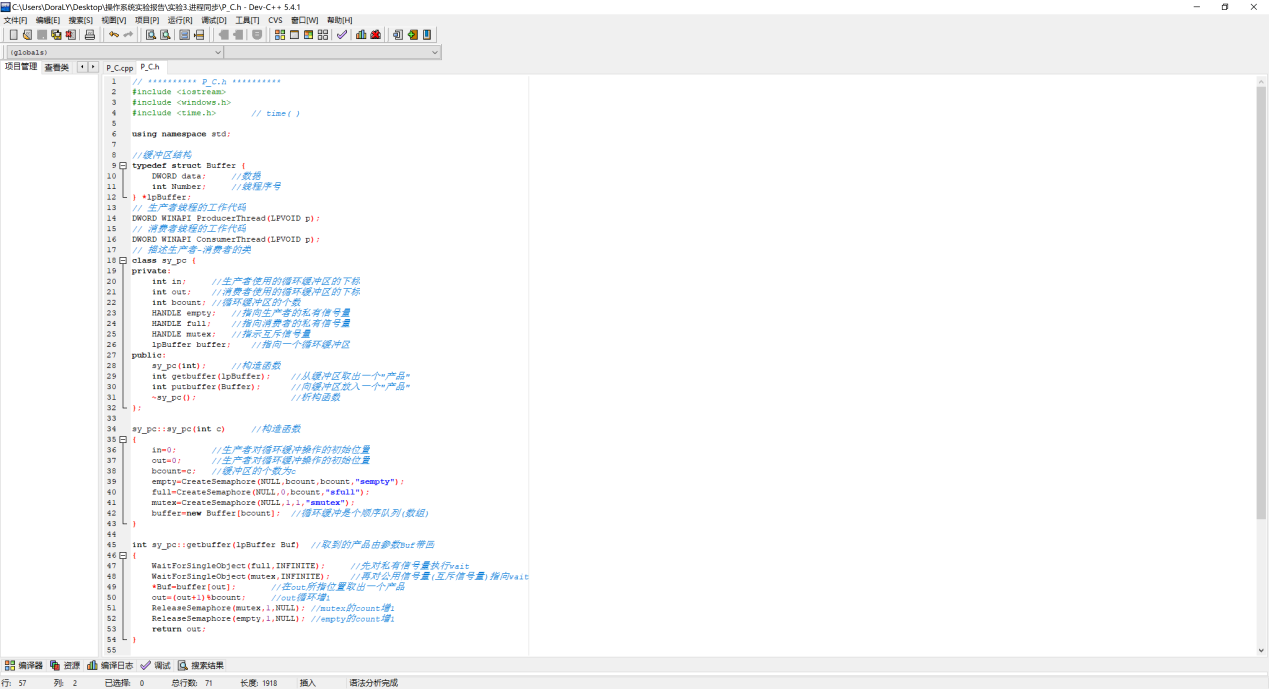
**4．CreateThread( ) //** 创建线程函数

**5．有关临界区(CRITICAL\_SECTION)的函数**

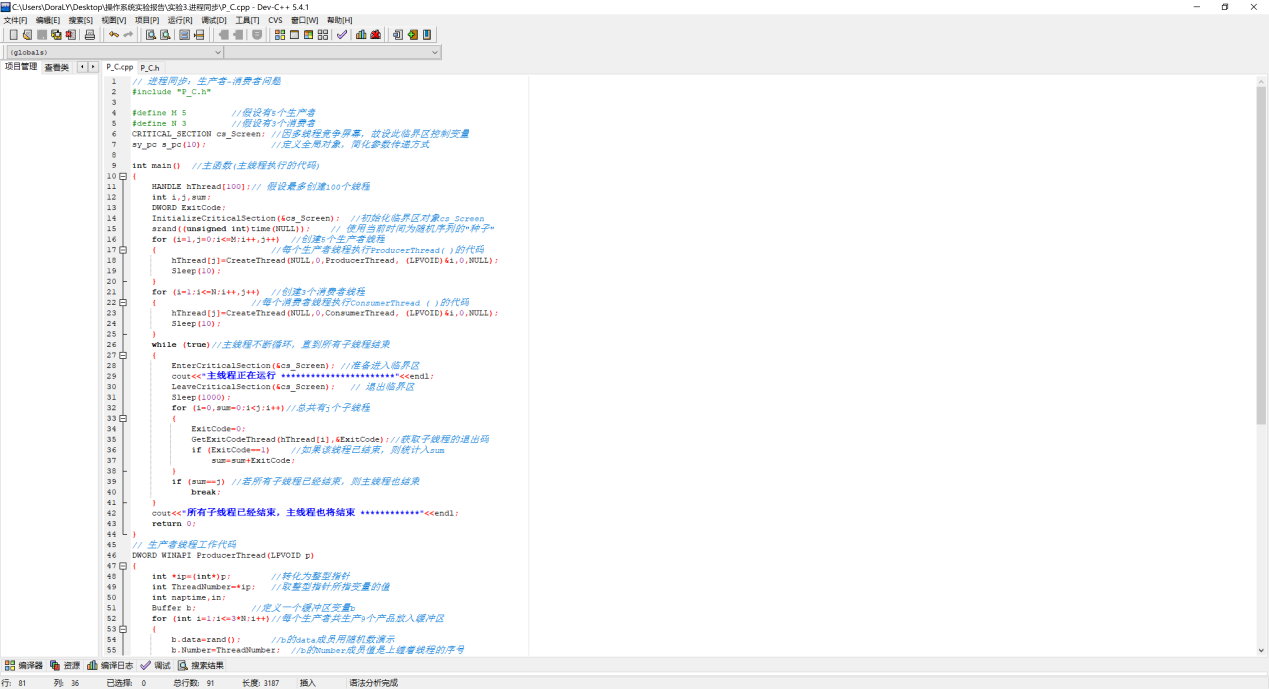
InitializeCriticalSection( )、EnterCriticalSection( )、LeaveCriticalSection( )，参看2.2.2节。

**6．Sleep( )**

1. **实验步骤：**
2. **编写“P\_C.h”**

****

1. **编写“P\_C.cpp”**

****

**3．编译运行**

1. **实验源程序**

**1.P\_C.h**

**// \*\*\*\*\*\*\*\*\*\* P\_C.h \*\*\*\*\*\*\*\*\*\***

**#include <iostream>**

**#include <windows.h>**

**#include <time.h> // time( )**

**using namespace std;**

**//缓冲区结构**

**typedef struct Buffer {**

**DWORD data; //数据**

**int Number; //线程序号**

**} \*lpBuffer;**

**// 生产者线程的工作代码**

**DWORD WINAPI ProducerThread(LPVOID p);**

**// 消费者线程的工作代码**

**DWORD WINAPI ConsumerThread(LPVOID p);**

**// 描述生产者-消费者的类**

**class sy\_pc {**

**private:**

**int in; //生产者使用的循环缓冲区的下标**

**int out; //消费者使用的循环缓冲区的下标**

**int bcount; //循环缓冲区的个数**

**HANDLE empty; //指向生产者的私有信号量**

**HANDLE full; //指向消费者的私有信号量**

**HANDLE mutex; //指示互斥信号量**

**lpBuffer buffer; //指向一个循环缓冲区**

**public:**

**sy\_pc(int); //构造函数**

**int getbuffer(lpBuffer); //从缓冲区取出一个"产品"**

**int putbuffer(Buffer); //向缓冲区放入一个"产品"**

**~sy\_pc(); //析构函数**

**};**

**sy\_pc::sy\_pc(int c) //构造函数**

**{**

**in=0; //生产者对循环缓冲操作的初始位置**

**out=0; //生产者对循环缓冲操作的初始位置**

**bcount=c; //缓冲区的个数为c**

**empty=CreateSemaphore(NULL,bcount,bcount,"sempty");**

**full=CreateSemaphore(NULL,0,bcount,"sfull");**

**mutex=CreateSemaphore(NULL,1,1,"smutex");**

**buffer=new Buffer[bcount]; //循环缓冲是个顺序队列(数组)**

**}**

**int sy\_pc::getbuffer(lpBuffer Buf) //取到的产品由参数Buf带回**

**{**

**WaitForSingleObject(full,INFINITE); //先对私有信号量执行wait**

**WaitForSingleObject(mutex,INFINITE); //再对公用信号量(互斥信号量)指向wait**

**\*Buf=buffer[out]; //在out所指位置取出一个产品**

**out=(out+1)%bcount; //out循环增1**

**ReleaseSemaphore(mutex,1,NULL); //mutex的count增1**

**ReleaseSemaphore(empty,1,NULL); //empty的count增1**

**return out;**

**}**

**int sy\_pc::putbuffer(Buffer b)**

**{**

**WaitForSingleObject(empty,INFINITE);**

**WaitForSingleObject(mutex,INFINITE);**

**buffer[in]=b;**

**in=(in+1)%bcount;**

**ReleaseSemaphore(mutex,1,NULL);**

**ReleaseSemaphore(full,1,NULL);**

**return in;**

**}**

**sy\_pc::~sy\_pc() //析构函数**

**{**

**delete []buffer;**

**}**

1. **P\_C.cpp**

**// 进程同步：生产者-消费者问题**

**#include "P\_C.h"**

**#define M 5 //假设有5个生产者**

**#define N 3 //假设有3个消费者**

**CRITICAL\_SECTION cs\_Screen; //因多线程竞争屏幕，故设此临界区控制变量**

**sy\_pc s\_pc(10); //定义全局对象，简化参数传递方式**

**int main() //主函数(主线程执行的代码)**

**{**

**HANDLE hThread[100];// 假设最多创建100个线程**

**int i,j,sum;**

**DWORD ExitCode;**

**InitializeCriticalSection(&cs\_Screen); //初始化临界区对象cs\_Screen**

**srand((unsigned int)time(NULL)); // 使用当前时间为随机序列的"种子"**

**for (i=1,j=0;i<=M;i++,j++) //创建5个生产者线程**

**{ //每个生产者线程执行ProducerThread( )的代码**

**hThread[j]=CreateThread(NULL,0,ProducerThread, (LPVOID)&i,0,NULL);**

**Sleep(10);**

**}**

**for (i=1;i<=N;i++,j++) //创建3个消费者线程**

**{ //每个消费者线程执行ConsumerThread ( )的代码**

**hThread[j]=CreateThread(NULL,0,ConsumerThread, (LPVOID)&i,0,NULL);**

**Sleep(10);**

**}**

**while (true)//主线程不断循环，直到所有子线程结束**

**{**

**EnterCriticalSection(&cs\_Screen); //准备进入临界区**

**cout<<"主线程正在运行 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**LeaveCriticalSection(&cs\_Screen); // 退出临界区**

**Sleep(1000);**

**for (i=0,sum=0;i<j;i++)//总共有j个子线程**

**{**

**ExitCode=0;**

**GetExitCodeThread(hThread[i],&ExitCode);//获取子线程的退出码**

**if (ExitCode==1) //如果该线程已结束，则统计入sum**

**sum=sum+ExitCode;**

**}**

**if (sum==j) //若所有子线程已经结束，则主线程也结束**

**break;**

**}**

**cout<<"所有子线程已经结束，主线程也将结束 \*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**return 0;**

**}**

**// 生产者线程工作代码**

**DWORD WINAPI ProducerThread(LPVOID p)**

**{**

**int \*ip=(int\*)p; //转化为整型指针**

**int ThreadNumber=\*ip; //取整型指针所指变量的值**

**int naptime,in;**

**Buffer b; //定义一个缓冲区变量b**

**for (int i=1;i<=3\*N;i++)//每个生产者共生产9个产品放入缓冲区**

**{**

**b.data=rand(); //b的data成员用随机数演示**

**b.Number=ThreadNumber; //b的Number成员值是上缠着线程的序号**

**in=s\_pc.putbuffer(b); //将b放入缓冲区，返回当前的下标in的值**

**EnterCriticalSection(&cs\_Screen); //准备进入临界区**

**cout<<"Producer "<<ThreadNumber<<" 向缓冲区投放了第 "**

**<<i<<" 个数据 "<<b.data<<"。in="<<in<<endl;**

**LeaveCriticalSection(&cs\_Screen); // 退出临界区**

**naptime=100+rand()%200; //睡眠100~300ms**

**Sleep(naptime);**

**}**

**EnterCriticalSection(&cs\_Screen); //准备进入临界区**

**cout<<"Producer "<<ThreadNumber<<" 运行完毕"<<endl;**

**LeaveCriticalSection(&cs\_Screen); // 退出临界区**

**return 1; //子线程的返回值(退出码)为1**

**}**

**// 消费者线程工作代码**

**DWORD WINAPI ConsumerThread(LPVOID p)**

**{**

**int ThreadNumber=\*((int\*)p);**

**int naptime,out;**

**Buffer b;**

**for (int i=1;i<=3\*M;i++)//每个消费者共获取15个产品后结束**

**{**

**out=s\_pc.getbuffer(&b);//从缓冲区取产品放在b中，返回下标out**

**EnterCriticalSection(&cs\_Screen); //准备进入临界区**

**cout<<"Consumer "<<ThreadNumber<<" 从缓冲区取得了第 "**

**<<i<<" 个数据 "<<b.data;**

**cout<<", 它是Producer "<<b.Number<<"存放的。out="<<out<<endl;**

**LeaveCriticalSection(&cs\_Screen); // 退出临界区**

**naptime=100+rand()%200;**

**Sleep(naptime); //睡眠等待一段时间只是为了演示**

**}**

**EnterCriticalSection(&cs\_Screen); //准备进入临界区**

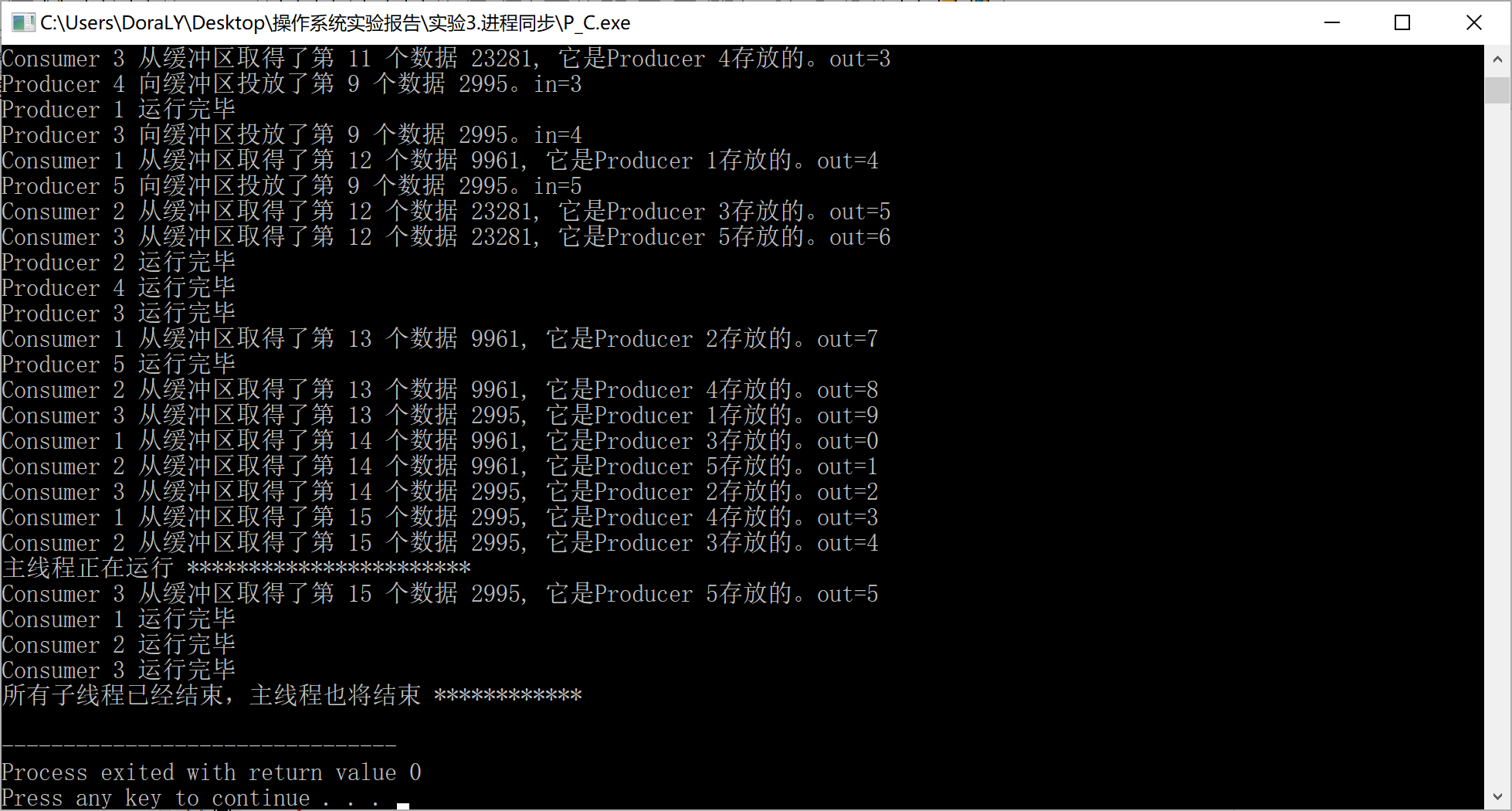
**cout<<"Consumer "<<ThreadNumber<<" 运行完毕"<<endl;**

**LeaveCriticalSection(&cs\_Screen); // 退出临界区**

**return 1; //子线程的返回值(退出码)为1**

**}**

1. **实验结果**

****

1. **实验总结：**

通过这次进程同步实验，我学习到了在Win32程序中利用操作系统提供的API创建和操作semaphore，并通过编写程序查看其运行情况来分析每个生产者和消费者的活动情况以及缓冲区的存储状态，加深对生产者-消费者问题的感性认识和理解。学会了他们的用法之后我觉得对我以后的代码生涯也大有益处！我在实验期间还详细查找了生产者-消费者问题相关资料，通过阅读这些资料，我也弄懂了许多之前没有弄懂的问题。这次成功的进程同步实验也增强了我的自信心。总而言之，这次实验也和之前的实验一样，拓宽了我的眼界，使我更加熟悉了代码编写方面的操作，我现在已经有了一个比较纯熟的处理手段，我相信这对我以后的工作生活一定会起到很大的帮助！