**西 南 石 油 大 学 实 验 报 告**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **课 程** | 操作系统 | **实验项目** | | | 进程同步演示 | | | | | **日期** | | 2019.11.3 | |
| **专业年级** | 17级软件工程 | | **学号** | **201731062208** | | **姓名** | **李文毅** | **教师** | **黄诚** | | **成绩** | |  |

**一、实验目的**

通过学习调用Windows进程同步API实现进程同步演示的方法，深入了解进线程同步与互斥的含义与编程方法。

**二、教学要求**

1、理解生产者和消费者同步问题，进一步掌握Windows下进线程的控制方法，掌握Windows进程同步机制—信号量机制，掌握相应的API函数及其使用方法；

2、并实现简单信号量的应用编程，理解Windows中多线程的并发机制，线程间的同步和互斥。

**三、实验过程**

1、1）生产者-消费者问题参考代码pc.cpp

// pc.cpp : 控制台应用程序的入口点。

#include "stdio.h"

#include <windows.h>

#include <iostream>

const unsigned short SIZE\_OF\_BUFFER = 10; //缓冲区长度

unsigned short ProductID = 0; //产品号

unsigned short ConsumeID = 0; //将被消耗的产品号

unsigned short in = 0; //产品进缓冲区时的缓冲区下标

unsigned short out = 0; //产品出缓冲区时的缓冲区下标

int g\_buffer[SIZE\_OF\_BUFFER]; //缓冲区是个循环队列

bool g\_continue = true; //控制程序结束

HANDLE g\_hMutex; //用于线程间的互斥

HANDLE g\_hFullSemaphore; //当缓冲区满时迫使生产者等待

HANDLE g\_hEmptySemaphore; //当缓冲区空时迫使消费者等待

DWORD WINAPI Producer(LPVOID); //生产者线程

DWORD WINAPI Consumer(LPVOID); //消费者线程

int main() {

//创建各个互斥信号

g\_hMutex = CreateMutex(NULL,FALSE,NULL);

g\_hFullSemaphore = CreateSemaphore ( NULL,

SIZE\_OF\_BUFFER - 1, SIZE\_OF\_BUFFER - 1, NULL);

g\_hEmptySemaphore = CreateSemaphore(NULL,0,SIZE\_OF\_BUFFER-1,NULL);

//调整下面的数值，可以发现，当生产者个数多于消费者个数时，

//生产速度快，生产者经常等待消费者；反之，消费者经常等待

const unsigned short PRODUCERS\_COUNT = 3; //生产者的个数

const unsigned short CONSUMERS\_COUNT = 1; //消费者的个数

//总的线程数

const unsigned short THREADS\_COUNT =

PRODUCERS\_COUNT+CONSUMERS\_COUNT;

HANDLE hThreads[THREADS\_COUNT]; //各线程的handle

DWORD producerID[PRODUCERS\_COUNT]; //生产者线程的标识符

DWORD consumerID[CONSUMERS\_COUNT]; //消费者线程的标识符

//创建生产者线程

for (int i=0;i<PRODUCERS\_COUNT;++i){

hThreads[i]=CreateThread(NULL,0,Producer,NULL,0,&producerID[i]);

if (hThreads[i]==NULL) return -1;

}

//创建消费者线程

for (i=0;i<CONSUMERS\_COUNT;++i){

hThreads[PRODUCERS\_COUNT+i]=

CreateThread(NULL,0,Consumer,NULL,0,&consumerID[i]);

if (hThreads[i]==NULL) return -1;

}

while(g\_continue){

if(getchar()){ //按回车后终止程序运行

g\_continue = false;

}

}

return 0;

}

//生产一个产品。简单模拟了一下，仅输出新产品的ID号

void Produce() {

std::cerr << "Producing " << ++ProductID << " ... ";

std::cerr << "Succeed" << std::endl;

}

//把新生产的产品放入缓冲区

void Append() {

std::cerr << "Appending a product ... ";

g\_buffer[in] = ProductID;

in = (in+1)%SIZE\_OF\_BUFFER;

std::cerr << "Succeed" << std::endl;

//输出缓冲区当前的状态

for (int i=0;i<SIZE\_OF\_BUFFER;++i){

std::cout << i <<": " << g\_buffer[i];

if (i==in) std::cout << " <-- 生产";

if (i==out) std::cout << " <-- 消费";

std::cout << std::endl;

}

}

//从缓冲区中取出一个产品

void Take() {

std::cerr << "Taking a product ... ";

ConsumeID = g\_buffer[out];

out = (out+1)%SIZE\_OF\_BUFFER;

std::cerr << "Succeed" << std::endl;

//输出缓冲区当前的状态

for (int i=0;i<SIZE\_OF\_BUFFER;++i){

std::cout << i <<": " << g\_buffer[i];

if (i==in) std::cout << " <-- 生产";

if (i==out) std::cout << " <-- 消费";

std::cout << std::endl;

}

}

//消耗一个产品

void Consume() {

std::cerr << "Consuming " << ConsumeID << " ... ";

std::cerr << "Succeed" << std::endl;

}

//生产者

DWORD WINAPI Producer(LPVOID lpPara) {

while(g\_continue){

WaitForSingleObject(g\_hFullSemaphore,INFINITE);

WaitForSingleObject(g\_hMutex,INFINITE);

Produce();

Append();

Sleep(1500);

ReleaseMutex(g\_hMutex);

ReleaseSemaphore(g\_hEmptySemaphore,1,NULL);

}

return 0;

}

//消费者

DWORD WINAPI Consumer(LPVOID lpPara) {

while(g\_continue){

WaitForSingleObject(g\_hEmptySemaphore,INFINITE);

WaitForSingleObject(g\_hMutex,INFINITE);

Take();

Consume();

Sleep(1500);

ReleaseMutex(g\_hMutex);

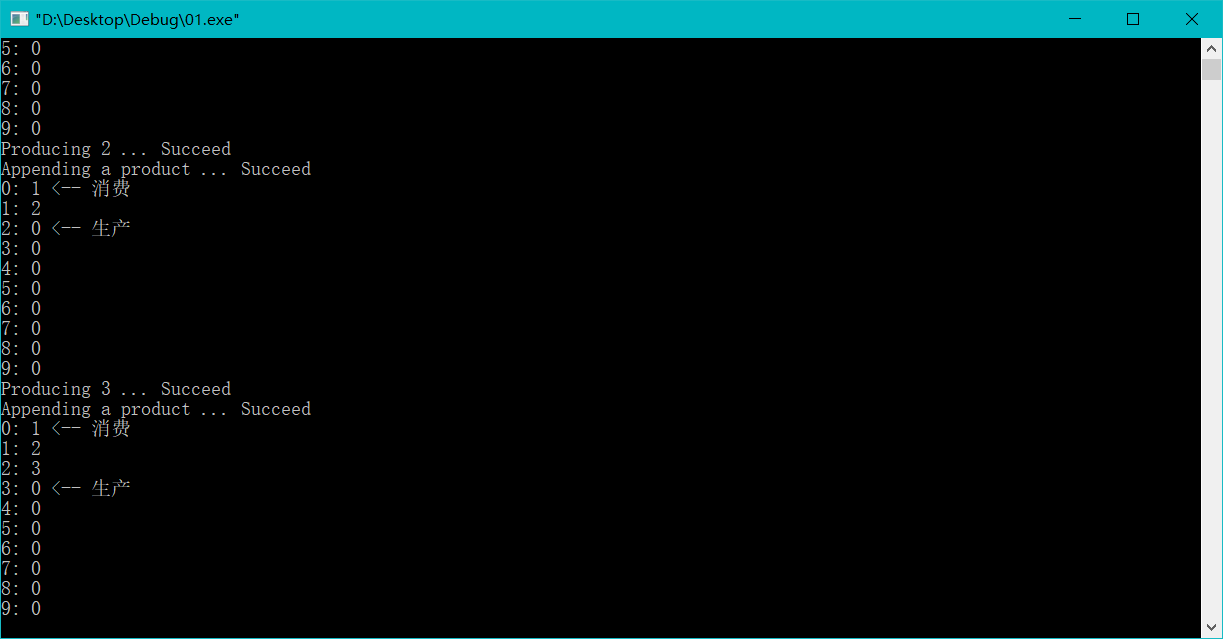
ReleaseSemaphore(g\_hFullSemaphore,1,NULL);

}

return 0;

}

3）运行结果截图



2、在1的基础上编写Windows下父亲儿子女儿放取水果进程同步的演示程序。

1）问题描述：桌上有一空盘，最多允许存放一个水果。爸爸可向盘中放一个苹果或放一个桔子，儿子专等吃盘中的桔子，女儿专等吃苹果。 试用P、V操作实现爸爸、儿子、女儿三个并发进程的同步。提示：设置一个信号量表示可否向盘中放水果，一个信号量表示可否取桔子，一个信号量表示可否取苹果。）

2）代码：

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <windows.h>

int plate = 0; //判断盘子是否为空

int SO = 0; //判断盘子里有几个桔子

int SA = 0; //判断盘子里有几个苹果

int S = 0; //判断盘子是否可以放水果

void ProcessFather()

{

while (plate < 1)

{

plate++;

S = 1;

srand((unsigned)time(NULL));

int a = rand() % 100000;

if (a % 2 == 0)

{

printf("father放了一个苹果\n");

SA++;

}

else

{

printf("father放了一个桔子\n");

SO++;

}

}

}

void ProcessSon()

{

if (S)

{

while (SO > 0)

{

Sleep(1000);

printf("son吃了一个桔子\n");

plate--;

SO--;

S = 0;

}

}

}

void ProcessDaughter()

{

if (S)

{

while (SA > 0)

{

Sleep(1000);

printf("daughter吃了一个苹果\n");

plate--;

SA--;

S = 0;

}

}

}

int main()

{

int i, n;

printf("father放了一次水果?\n");

scanf("%d", &n);

for (i = 0; i < n; i++)

{

ProcessFather();

ProcessSon();

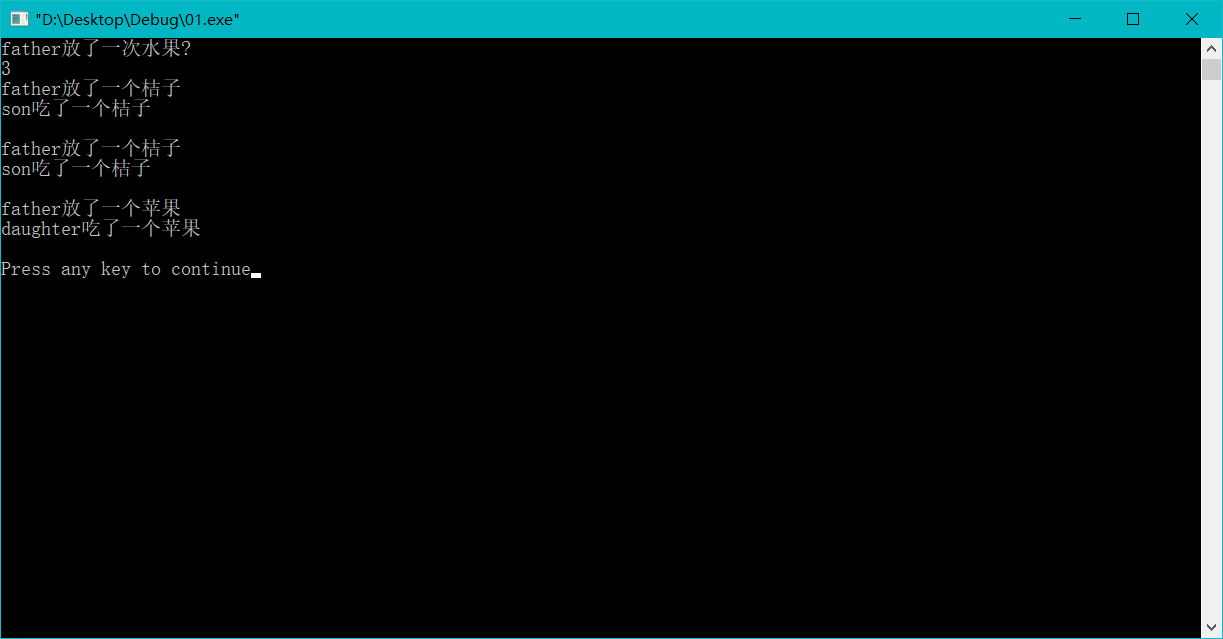
ProcessDaughter();

printf("\n");

}

}

3）运行结果截图



**四、实验总结**

通过这次实验我对信号量机制更加了解了。