 ****

**计算机科学与技术学院**

**《操作系统》实验报告**

（ 2019 / 2020 学年 第 二 学期）

**学生姓名： XXX**

**学生专业： 计算机科学与技术**

**学生班级： 计算机172002**

**学生学号： 201720XXXXX**

20 20 年　6 月　1 日

# 实验二 windows并发多线程的应用

实验学时：2

实验类型：验证

实验要求：必修

**一、实验目的**

本课题实习的目的是，加深对wiundows线程概念及线程同步管理各部分内容的理解；熟悉windows线程管理API的使用

**二、实验要求**

1.通过上网查阅资料，了解windows线程同步函数，写出本程序中出现的API函数的定义；

2.阅读案例程序，给出程序的详细注解；

3.运行程序，分析程序结果。

4. 改写程序实现（至少选其中一个项目实现）

（1）《苹果桔子问题》：

桌上有一只盘子，每次只能放入一只水果；爸爸专向盘子中放苹果(apple)，妈妈专向盘子中放桔于(orange)，一个儿子专等吃盘子中的桔子，一个女儿专等吃盘子里的苹果。

（2）设有线程A，B，C，分别调用过程get，copy和 put 对缓冲区S和T进行操作。其中get负责把数据块输入缓冲区S，copy负责从S中提取数据块并复制到缓冲区T中，put负责从缓冲区T中取出信息打印（如图所示）。编程序实现get，copy和 put并发的操作过程。

（3）编程序实现生产者—消费者问题。

5. 撰写上机报告。

**三、实验分析**

我实现的是生产者消费者问题。也称有限缓冲问题，描述了两个共享固定大小缓冲区的线程——即所谓的“生产者”和“消费者”——在实际运行时会发生的问题。生产者的主要作用是生成一定量的数据放到缓冲区中，然后重复此过程。与此同时，消费者也在缓冲区消耗这些数据。该问题的关键就是要保证生产者不会在缓冲区满时加入数据，消费者也不会在缓冲区中空时消耗数据。

如何有效的解决生产者消费者问题？

要解决该问题，可以选择两者同一时间只能出现一个，但是第一个出现的必须是生产者。为此引入两类线程，分别代表生产者producer和消费者consumer，生产者线程用于生产数据，消费者线程用于消费数据。为了解耦生产者和消费者的关系，通产会采用共享的数据区域，生产者生产数据之后直接放置在共享数据区中，并不需要关心消费者的行为：而消费者只需要从共享数据区中去获取数据，就不需要关系生产者的行为。但是这个共享数据区应该具备这样的线程间并发协作的功能：

1. 共享数据区buffer处于满状态（T），阻塞生产者继续生产数据。
2. 共享数据区buffer处于空状态（T），阻塞消费者继续使用数据。

信号量

信号量又称为信号标，是一个同步对象，用于保持在0至指定最大值之间的一个计数值。当线程完成一次对该semaphore对象的等待（wait）时，该计数值减一；当线程完成一次对semaphore对象的释放（release）时，计数值加一。当计数值为0，则线程等待该semaphore对象不再能成功直至该semaphore对象变成signaled状态。semaphore对象的计数值大于0，为signaled状态；计数值等于0，为nonsignaled状态.

初始化，给与它一个非负数的整数值。

运行P（wait()），信号标S的值将被减少。企图进入临界区段的进程，需要先运行P（wait()）。当信号标S减为负值时，进程会被挡住，不能继续；当信号标S不为负值时，进程可以获准进入临界区段。

运行V（signal()），信号标S的值会被增加。结束离开临界区段的进程，将会运行V（signal()）。当信号标S不为负值时，先前被挡住的其他进程，将可获准进入临界区段。

P操作：使 S=S-1 ，若 S>=0 ，则该进程继续执行，否则排入等待队列。

V操作：使 S=S+1 ，若 S>0 ,唤醒等待队列中的一个进程。

**四、实验代码**

#include<stdio.h>

#include<windows.h>

//利用数组模拟缓冲区

#define SIZE 3 //缓冲区容量大小

typedef int semaphore; //信号量

mutex =1 : 表示没有进程进入临界区

mutex =0 : 表示有一个进程进入临界区，前方没有其他进程。

mutex =-1 :表示一个进程已经进入，另一个进程的等待进入。

semaphore mutex = 1;//互斥锁

semaphore empty = SIZE;//缓冲区中未使用个数

semaphore full = 0;//缓冲区中已使用个数

semaphore buffer[SIZE];//创建缓冲区

semaphore front = 0, rear = 0;

void producer\_Therad();//生产者进程

void consumer\_Thread();//消费者进程

void P(semaphore& pass) {// P操作 passeren 荷兰语 通过

pass--;

}

void V(semaphore& release) {// V操作 vrijgevern 荷兰语 释放

release++;

}

void produce\_item(char\* item\_ptr) {//生产者生产食物 并标记为T True 代表该处已满

\*item\_ptr = 'T';

}

void put\_buffer(int pass) {//生产者将物品放入缓冲区

buffer[front] = pass;

printf("produce is %c into buffer[%d]\n", buffer[front], front);

front = (front + 1) % SIZE;

}

void remove\_buffer(char\* release) {//消费者消费缓冲区的食物，并标记为F false 代表该处已空

printf("consumer use is %c from buffer[%d]\n", buffer[rear], rear);

\*release = buffer[rear];

buffer[rear] = 'F';

printf("now the buffer[%d] is %c !\n", rear, buffer[rear]);

rear = (rear + 1) % SIZE;

}

void consume\_item() {

printf("consumer had use the product\n\n");

}

void producer\_Thread()

{

char item;

while (1) {

Sleep(1000);

produce\_item(&item);

P(empty);

P(mutex);

put\_buffer(item);

V(mutex);

V(full);

if (full == SIZE)

consumer\_Thread();

}

}

void consumer\_Thread()

{

char get\_item;

while (1) {

Sleep(1000);

P(full);

P(mutex);

remove\_buffer(&get\_item);

V(mutex);

V(empty);

consume\_item();

if (empty == SIZE)

producer\_Thread();

}

}

int main()

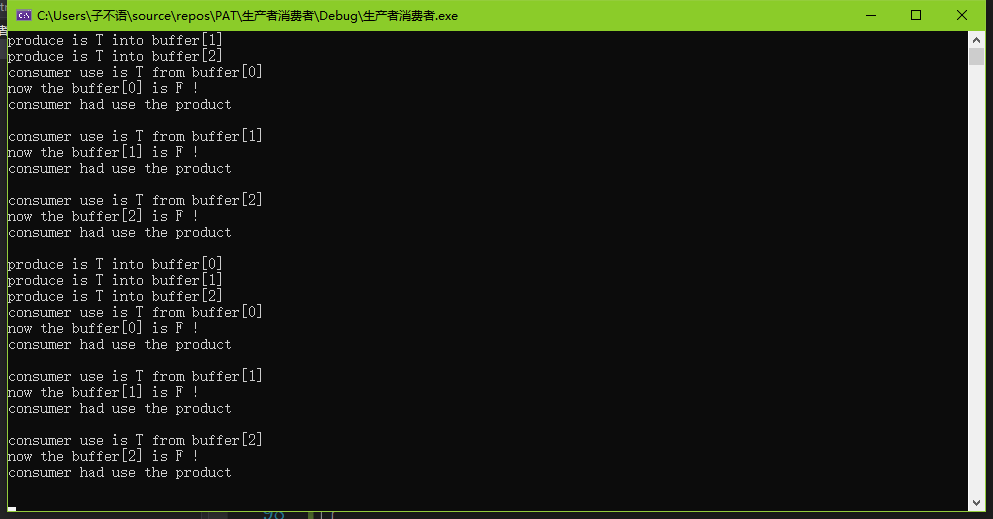
{

producer\_Thread();

return 0;

}

1. **实验结果**



1. **实验总结**

在这次实验中，通过对生产者和消费者的问题的实验，我对进程间的同步和异步机制有了更加深刻的认识，也对Windows系统中线程级应用的实现有了更深的了解。我也了解了使用来信号量的控制实现进程的同步和异步这种方法。在实现实验的过程中，还是要注意设置缓冲区的大小和使用规则，这一点特别容易出错，我在这里遇到了一定的困难，最后在查阅资料之后解决了这里的问题。