电子科技大学

计算机专业类课程

实验报告

课程名称：操作系统

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：冯铭扬

学　　号：2013060109023

指导教师：薛瑞尼

日　　期：2016　年　6　月　10　日

电 子 科 技 大 学

实　　验　　报　　告

实验一

一、实验名称：生产者消费者

二、实验学时：4

三、实验内容和目的：共享缓冲区中放置一个数字，取值范围为[0, 10]，初值为0。生产者将此值加1，消费者将此值减1。

1 场景1

- 同一进程内启动一组生产者线程和一组消费者线程
- 缓冲区为本进程的全局变量

2 场景2

- 启动一组生产者进程和一组消费者进程
- 同一个数据文件为缓冲区

• 输入

- p：生产者数量
- c：消费者数量

输出打印当前共享缓冲区中的数值，或者生产者消费者的状态。

四、实验器材（设备、元器件） windows ＋ vc＋＋

五、实验数据及结果分析：

实验代码

```
#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[PRODUCERS_COUNT]; //各线程的handle

    DWORD producerID[CONSUMERS_COUNT]; //生产者线程的标识符

    DWORD consumerID[THREADS_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,&consum
erID[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;
}
```