北 京 林 业 大 学

2019 学年—2020 学年第 2 学期操作系统实验报告书

专业：计算机科学与技术(创新实验班)　班级：计创 18

姓　　名：连月菡　　学　　号：181002222

实验地点：家　　任课教师：李巨虎

实验题目：实验 2　生产者与消费者（综合性实验）

实验环境：Visual Studio 2019 Community

一、 实验目的

通过实验模拟生产者与消费者之间的关系，了解并掌握他们之间的关系及其原理。由此增加对进程同步的问题的了解。

二、 实验内容

① 由用户指定要产生的进程及其类别，存入进入就绪队列。
② 调度程序从就绪队列中提取一个就绪进程运行。如果申请的资源被阻塞则进入相应的等待队列，调度程序调度就绪队列中的下一个进程。进程运行结束时，会检查对应的等待队列，激活队列中的进程进入就绪队列。运行结束的进程进入 over 链表。重复这一过程直至就绪队列为空。
③ 程序询问是否要继续？如果要转直①开始执行，否则退出程序。

三、 实验要求

每个进程有一个进程控制块（PCB）表示。进程控制块可以包含如下信息：进程类型标号、进程系统号、进程状态、进程产品（字符）、进程链指针等等。

系统开辟了一个缓冲区，大小由 buffersize 指定。

程序中有三个链队列，一个链表。一个就绪队列（ready），两个等待队列：生产者等待队列（producer）；消费者等待队列（consumer）。一个链表（over），用于收集已经运行结束的进程

本程序通过函数模拟信号量的原子操作。

四、 实验结果

实验代码如下:

头文件与结构体

```
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
using namespace std;
#define buffersize 3

typedef struct PCB {
    int name;        //进程名
    int state;       //状态
    bool attribute;      //属性, 属于生产者:0 属于消费者:1

}pcb;

typedef struct QNode {
    PCB   data;
    struct QNode* next;
}QNode, * QueuePtr;
typedef struct {
    QueuePtr  front;            //队列头
    QueuePtr  rear;             //队列尾
}LinkQueue;
```

全局变量

```
LinkQueue wait, ok, producer,consumer,over;//就绪, 完成, 生产者, 消费者队列
char buffer[buffersize];        //缓冲区:填进程name
int in = 0;                     //放入缓冲区指针
int out = 0;                        //取走缓冲区指针
int productnum = 0;                //产品数
int full = 0;                       //信号量
int emptys = buffersize;            //信号量
int mutex = 1;                  //互斥信号量
```

队列的基本操作，初始化，进队出队等

```
int InitQueue(LinkQueue& Q) {//初始化队列
    Q.front = Q.rear = new QNode;
    if (!Q.front)
        exit(0);
    Q.front->next = NULL;
    return 1;
}
int EnQueue(LinkQueue& Q, PCB e) {//进队
    QueuePtr p;
    p = new QNode;
    if (!p)
        exit(0);
    p->data = e;
    p->next = NULL;
    Q.rear->next = p;
    Q.rear = p;
    return 1;
}

int DeQueue(LinkQueue& Q, PCB& e) {//出队
    QueuePtr p;
    p = (QueuePtr)malloc(sizeof(QNode));
    if (Q.front == Q.rear)
        return 0;
    p = Q.front->next;
    e = p->data;
    Q.front->next = p->next;
    if (Q.rear == p)
        Q.rear = Q.front;
    free(p);
    return 1;
}

int QueueEmpty(LinkQueue Q) {//判断队列是否为空
    if (Q.front == Q.rear)
        return 0;
    else
        return 1;
}
```

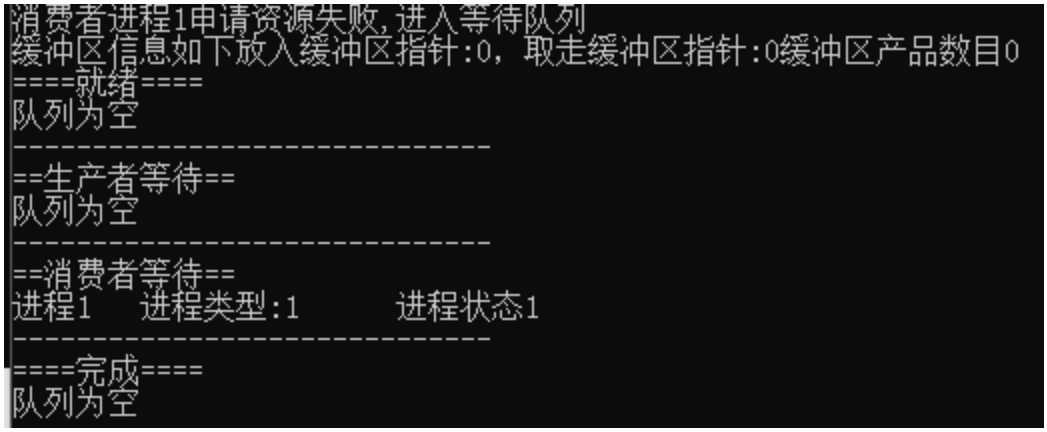
P/V操作,唤醒消费者生产者的函数

```
int P(int& s)                    //P操作
{
    return  s >= 1?--s : -1;
}

int V(int& s)                    //V操作
{
    return ++s;
}

void wake_con()                        //唤醒消费者
{
    if (consumer.front->next != NULL)           //队列不空才能唤醒
    {
        PCB p = consumer.front->next->data; //等待队头出进程
        DeQueue(consumer, p);
        EnQueue(ok, p);
        p.state = 1;
        cout << "唤醒消费者进程, " << p.name<< ", 插入就绪队列" << endl;
    }
    else
        cout << "消费者等待队列为空" << endl;
}
void wake_pror()                          //唤醒消费者
{
    if (producer.front->next != NULL)           //队列不空才能唤醒
    {
        PCB  p = producer.front->next->data;
        DeQueue(producer, p);
        EnQueue(ok, p);
        p.state = 1;
        cout << "唤醒生产者进程" << p.name << ", 插入就绪队列" << endl;
    }
    else
        cout << "生产者等待队列为空" << endl;
}
```

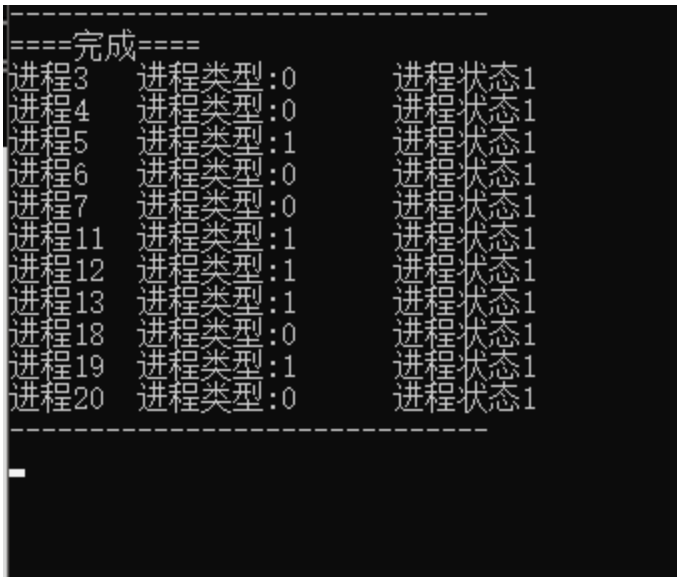
```
int main() {
	bool go;
	do{
		system("cls");
		InitQueue(ok); InitQueue(wait); InitQueue(producer); InitQueue(consumer); InitQueue(over);//初始化
队列		cout << "进程" << "\t" << "状态" << "\t" << "类型" << endl;
		for (int i = 0; i < 20; ++i)//创建进程
		{
			PCB pcb;
			pcb.name = i + 1;
			pcb.state = 1;
			pcb.attribute= rand() % 2;
			cout << pcb.name << "\t" <<   pcb.state << "\t" <<   pcb.attribute << endl;
			EnQueue(wait, pcb);
		}

		while (wait.front->next!=NULL) {
			QNode* pcb = wait.front->next;
			if (pcb->data.attribute == 0)//生产者
			{
				if (P(emptys) >= 0 && P(mutex) >= 0)			//P操作申请临界资源，继续执行本进程，或
者等待
				{
					buffer[in] = pcb->data.name;
					productnum++;					//缓冲区产品数目加1
					in++;							//放入缓冲区指针加1
					in = in % buffersize;
					EnQueue(over, pcb->data);
					pcb->data.state = 0;					//修改进程状态
					cout << "生产者进程" << pcb->data.name << "申请资源成功,产品总数为" << productnum
<< ",进入完成队列" << endl;
					V(mutex);
					V(full);						//V操作
				 wake_con();					//唤醒消费者
				}
				else							//等待
				{
					cout << "生产者进程" << pcb->data.name<< "申请资源失败,进入等待队列" << endl;
					EnQueue(producer,pcb->data);				//生产者队尾入进程
					pcb->data.state = 1;					//修改进程状态
				}
			}
			else							//消费者进程
			{
				if (P(full) >= 0 && P(mutex) >= 0)				//P操作，继续执行本进程，否
则本进程等待
				{
					int goods;
					goods = buffer[out];
					productnum--;					//数目减1
					out++;							//指针加1
					out = out % buffersize;
					EnQueue(over,pcb->data);				//完成队尾入进程
					pcb->data.state = 0;					//修改进程状态
					cout << "消费者进程" << pcb->data.name<< "申请资源成功,产品总数为" << productnum
<< ",进入over队列" << endl;
					V(mutex);
					V(emptys);						//V操作
					wake_pror();					//唤醒生产者
				}
				else{							//等待
					cout << "消费者进程" << pcb->data.name << "申请资源失败,进入等待队列" << endl;
					EnQueue(consumer,pcb->data);				//消费者队尾入进程
					pcb->data.state= 1;				//修改进程状态
				}
			}
			DeQueue(wait, pcb->data);
			cout << "缓冲区信息如下" << "放入缓冲区指针:" << in << ", 取走缓冲区指针:" << out << "缓冲区产
品数目" << productnum << endl;
			cout << "====就绪====" << endl;show(ok.front);
			cout << "==生产者等待==" << endl;show(producer.front);
			cout << "==消费者等待==" << endl;show(consumer.front);
			cout << "====完成====" << endl;show(over.front);
		}
		cin >> go;
	}while (go);
	return 0;
}
```

点击运行

就绪队列为空时，进程调度结束，此时各进程信息如下。



运行结束，输入 1 继续下一轮，输入 0 退出程序



====完成====
进程3 进程类型:0 进程状态1
进程4 进程类型:0 进程状态1
进程5 进程类型:1 进程状态1
进程6 进程类型:0 进程状态1
进程7 进程类型:0 进程状态1
进程11 进程类型:1 进程状态1
进程12 进程类型:1 进程状态1
进程13 进程类型:1 进程状态1
进程18 进程类型:0 进程状态1
进程19 进程类型:1 进程状态1
进程20 进程类型:0 进程状态1
