实验二 生产者-消费者问题

班级： 07112002 学号： 1120201316 姓名： 金雅各

一、实验目的

独立设计并编写程序实现生产者-消费者问题，加深对进程通信的理解。

二、实验内容

1. 在Windows系统下实现生产者-消费者问题。

2. 具体要求如下：

(1) 创建一个包含6个缓冲区的缓冲池，初始为空，每个缓冲区能存放一个长度为10个字符的字符串。

(2) 创建2个生产者进程，要求每个生产者进程：

随机等待一段时间后，向缓冲区中添加数据；

若缓冲区已满，则等待消费者取走数据后再继续添加数据；

重复添加数据12次。

(3) 创建3个消费者进程，要求每个消费者进程：

随机等待一段时间后，从缓冲区中读取数据；

若缓冲区为空，则等待生产者添加数据后再继续读取数据；

重复读取数据8次。

(4) 输出显示每次添加或读取数据的时间，以及缓冲区的映像。

三、实验步骤

① 编写实现生产者-消费者问题的C语言程序

使用win32api的线程库来实现线程的创建和管理

定义宏来确定生产者、消费者线程的数目，缓冲区的个数以及源数据数量

# define bufferSize 7 //缓冲区个数，实际为 bufferSize -1

# define threadProducer 2 //生产者线程为 2

# define threadConsumer 3 //消费者线程为 3

# define contentSize 12 //每个生产者要生产的产品数量

定义缓冲区结构体Buffer（即循环队列） 用以存储缓冲区以及头尾指）

typedef struct{

// 定义缓冲池，每个缓冲区大小为10个的字符串 即大小为size的循环队列

char\* buffer[bufferSize];

// 用以存取缓冲区的指针 当rear==front时缓冲区为空，当(rear+1)%size==front%size时 缓冲区为满

int front,rear;

}Buffer;

定义存储线程信息的结构体ThreadInfo 用以存储线程HANDLE与ID

typedef struct{

HANDLE handle; // 存储线程的HANDLE

DWORD id; // 存储线程的ID

}ThreadInfo;

定义数据源结构体Psrc 存储生产者要传入缓冲区的内容,以及产品数量

// 定义数据源结构体

typedef struct{

char\* src[contentSize];

int size;

}Psrc;

定义等待时间，包含阻塞等待时间以及线程等待时间

DWORD WaitTime=INFINITE; //阻塞等待时间

DWORD ProdeucerWaitTime=1000; //生产者向缓冲区送入产品后的等待时间

DWORD ConsumerWaitTime=3000; //消费者从缓冲区取出产品后的等待时间

定义信号量，包含同步和互斥信号量

// 定义信号量

HANDLE mutex; //生产者与消费者对于缓冲区的互斥信号量

HANDLE empty; //缓冲区剩余数量的同步信号量

HANDLE full; //缓冲区中产品个数的同步信号量

主要使用的函数有：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | CreateSemaphore | | | |
| 返回值 | HANDLE | | | |
| 参数 | LPSECURITY\_ATTRIBUTES | LONG | LONG | LPCSTR | |
|  | 指定一个SECURITY\_ATTRIBUTES结构，或传递零值 | 信号量对象的初始计数 | 信号量的最大值 | 信号量对象的名称。 | |
| 函数作用 | 该函数返回同步信号量对象的句柄 | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | CreateMutex | | |
| 返回值 | HANDLE | | |
| 参数 | LPSECURITY\_ATTRIBUTES | BOOL | LPCTSTR | |
|  | 指定一个SECURITY\_ATTRIBUTES结构，或传递零值 | 初始化互斥对象的所有者。 | 指向互斥对象名的指针，设置互斥对象的名字。 | |
| 函数作用 | 该函数返回异步信号量对象的句柄 | | | |

|  |  |
| --- | --- |
| 函数名 | Producer |
| 返回值 | DWORD WINAPI |
| 参数 | LPVOID | |
|  | 传入线程需要处理的数据，这里传入的是生产者要放入缓冲区中的源数据结构Psrc | |
| 函数作用 | 该函数指定到线程构造函数中，用以在线程中执行该函数 | |

|  |  |
| --- | --- |
| 函数名 | Consumer |
| 返回值 | DWORD WINAPI |
| 参数 | LPVOID | |
|  | 消费者线程无需传入参数 | |
| 函数作用 | 该函数指定到线程构造函数中，用以在线程中执行该函数 | |

|  |  |  |
| --- | --- | --- |
| 函数名 | pushBuffer | |
| 返回值 | void | |
| 参数 | char \* | Buffer \* | |
|  | 生产者要放入缓冲区中的“产品” | 缓冲区 | |
| 函数作用 | 该函数将指定产品输送到缓冲区中并打印系统时间以及显示此时缓冲区的映射 | | |

|  |  |
| --- | --- |
| 函数名 | popBuffer |
| 返回值 | char \* |
| 参数 | Buffer \* | |
|  | 缓冲区 | |
| 函数作用 | 该函数将从缓冲区中取出一个产品，并打印系统时间以及显示此时缓冲区的映射，最后返回产品； | |

|  |  |
| --- | --- |
| 函数名 | printNowTime |
| 返回值 | Void |
| 参数 | void | |
|  |  | |
| 函数作用 | 打印系统当前时间 | |

|  |  |
| --- | --- |
| 函数名 | showBuffer |
| 返回值 | Void |
| 参数 | Buffer\* | |
|  | 缓冲区 | |
| 函数作用 | 打印出缓冲区中剩余的“产品” | |

|  |  |
| --- | --- |
| 函数名 | main |
| 返回值 | Void |
| 参数 |  | |
|  |  | |
| 函数作用 | 进行数据的初始化（缓冲区，信号量，线程，线程信息）；  挂起主线程以执行 | |

源代码在文件 lab\_02.c 中

② 对程序进程测试

通过改变生产者与消费者的等待时间，观察缓冲区的使用情况；

四、实验结果及分析

本实验生产者线程为2，消费者线程为3

当生产者等待时间为 1000ms，消费者等待时间为3000ms时，会出现生产者等待缓冲区满的现象。

当消费者等待时间为3000ms，生产者等待时间为1000ms时，会出现消费者等待缓冲区满的现象。

由于生产者与消费者公平竞争使用缓冲区，且消费者数量为生产者数量的1.5倍。当生产者等待时间更短时，生产产品的效率更高，而消费的速率过慢，会出现缓冲区满的情况。当消费者等待时间更短时，消费产品的效率更高，产能跟不上，会出现消费者等待空缓冲区的情况

五、实验收获与体会

通过本次实验，对于windows系统使用线程的流程有了基本的了解，同时对于进程的并发控制和死锁有了更深的理解。当线程数以及等待时间不同时，同样的代码也会有不同的性能体现。在实际中，我们要根据每个生产者与每个消费者的效率，通过控制生产者与消费者的数量，尽量使生产者的总体性能与消费者的总效率相当，避免出现缓冲区空或满的情况。

附录：程序清单及说明（黑体四号字）

宋体小四号字，行距22磅

（列出文件名及说明即可，不需要在此处复制代码，代码直接以源文件形式提供，但源文件中对代码要有必要的注释和说明）