P操作：passeren，译为"通过“

2）若sem>=0，则P操作返回，该线程程可以“通过”并继续执行。

3）若sem<0，则该线程被阻塞，进入操作系统的阻塞队列。

1）sem加1（释放）。  
2）若sem>0，则V操作返回，该线程继续执行。  
3）若sem<=0，则从阻塞队列中唤醒一个阻塞在该信号量上的线程，然后再返回原线程（调用ν操作的线程）继续执行。

生产者消费者问题（Producer-consumer problem）是一个多线程经典案例，该问题描述了两个线程（“生产者”和“消费者”）使用块缓冲区。生产者生成数据放到缓冲区中，消费者在缓冲区取出数据。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空时取出数据。

#include<windows.h>

#include<queue>

#include <process.h>

using namespace std;

//有一个仓库

queue<int> store;

int StoreSize = 3;// 仓库可以放3个货物

int ID = 1;//货物起始ID

//随机时间数组，模拟随机生产与消费的速度

int arr1[10] = { 2,1,4,0,3,4,6,1,9,0};

int arr2[10] = { 1,9,5,4,9,3,4,0,2,6};

//需要两个event来通知

HANDLE hEvent1 = INVALID\_HANDLE\_VALUE;//有货物时通知消费者取货物

HANDLE hEvent2 = INVALID\_HANDLE\_VALUE;//仓库为空通知消费者开始生产

//生产者

void ProducerThread(LPVOID param);

//消费者

void ConsumerThread(LPVOID param);

int main()

{

hEvent1 = CreateEvent(NULL, TRUE, TRUE, L"事件对象1");//需要先开始生产

hEvent2 = CreateEvent(NULL, TRUE, FALSE, L"事件对象2");//一开始，仓库没货取

uintptr\_t t1= \_beginthread(ProducerThread, 0, NULL);

uintptr\_t t2 = \_beginthread(ConsumerThread, 0, NULL);

//无限等待两个线程运行结束

HANDLE hArr []= { (HANDLE)t1 ,(HANDLE)t2 };

WaitForMultipleObjects(2, hArr, TRUE, INFINITE);

CloseHandle(hEvent1);

CloseHandle(hEvent2);

return 0;

}

//生产者

void ProducerThread(LPVOID param)

{

while (TRUE)

{

//看event是否允许生产

WaitForSingleObject(hEvent1, INFINITE);

if (store.size() < StoreSize)//有仓库空位才生产

{

int id = ID++;//货物ID号

printf("生产货物: %d\n" , id);

store.push(id);//把货物放置到仓库中

Sleep( arr1[id%10] \*1000 );//生产有时快，有时慢

}

else //仓库满了,需要停止生产

{

ResetEvent(hEvent1);//把事件设为无信号状态

printf("仓库满了！\n");

}

//仓库有货物,可以通知消费者取货物

if (store.size() > 0)

{

SetEvent(hEvent2);//让消费者的事件对象为有信号

}

}

}

//消费者

void ConsumerThread(LPVOID param)

{

while (TRUE)

{

//看event2是否允许取货物

WaitForSingleObject(hEvent2, INFINITE);

if (store.size() > 0)

{

int id = store.front();//获取队列中的货物，先进先出

store.pop();//卸走货物，腾出仓库空位

printf(" 取出货物: %d\n", id);

Sleep(arr2[id % 10] \* 1000);

}

else //仓库空了

{

ResetEvent(hEvent2);//设为无信号，不能取货物了

printf(" 仓库空了！\n");

}

if (store.size() < 3)//说明仓库没有满，可以继续生产

{

SetEvent(hEvent1);//通知生产者继续生产货物

}

}

}