|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 数学与信息科学学院 | | | | | | |  |
|  |  | | | | | | |  |
|  |  | | | | | | |  |
|  | **实验报告** | | | | | | |  |
|  |  | | | | | | |  |
|  |  | | | | | | |  |
|  | **课程名称：** | | **操作系统** | | | | |  |
|  | **姓 名：** | | **任希恒** | | | | |  |
|  | **学 号：** | | **541910010217** | | | | |  |
|  | **专业班级：** | | **信息与计算科学专业19-02班** | | | | |  |
|  | **指导教师：** | | **时海亮** | | | | |  |
|  |  | |  | | | | |  |
|  |  | |  | | | | |  |
|  |  | **2021-2022** | | **学年第** | **2** | **学期** |  |  |

**实验 1 线程的创建与撤销**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验日期：** | **2022年04月11日** | **实验类型：** | **验证型** | **实验成绩：** |  |

**一、实验目的**

通过本实验熟悉 Windows系统提供的线程创建与撤销等系统调用，掌握Windows系统环境下线程的创建与撤销方法。

**二、实验内容**

1．熟悉开发环境Visual C++ 6.0;

2. Windows系统环境下线程的创建与撤销方法;

3.利用2个线程模拟火车票销售系统。

**三、实验准备知识**

**1.线程创建函数**

HANDLE CreateThread (

LPSECURITY\_ATTRIBUTES lpThreadAttributes,

SIZE\_T dwstacksize,

LPTHREAD\_START\_ROUTINE lpstartAddress,LPVOID

lpParameter,

DWORD dwCreationFlags,

LPDWORD lpThreadId

）；

**1.线程撤销函数**

VOID ExitThread (

DWORD dwExitcode

**3.线程终止函数**

BOOLTerminateThread (

HANDLE hThread,

DWORD dwExitcode

**4.线程挂起函数**

VoID sleep(

DWORD dwMilliseconds

）；

**5．关闭句柄函数**

BOOL closeHandle(

HANDLE hobject

）；

**四、实验步骤**

**1. Windows系统环境下线程的创建与撤销方法**

该程序段在主线程中调用CreateThread()创建2个子线程，并在子线程中显示类似“Thread is running !”等字样。其主要代码如下:

#include <windows.h>

#include <iostream.h>

int index = 0;

DWORD WINAPI Thread1Proc(

LPVOID lpParameter

) ;

DWORD WINAPI Thread2Proc(

LPVOID lpParameter

);

int main ()

{

HANDLE hThreadl;

HANDLE hThread2;

hThreadl=CreateThread (NULL, //创建线程1

0,

Thread1Proc,

NULL,

0,

NULL);

hThread2=CreateThread(NULL, //创建线程2

0,

Thread2Proc,

NULL,

0,

NULL);

CloseHandle (hThread1); //关闭句柄

CloseHandle (hThread2);

while (index++<1000)

{

cout<<"main thread is running ! "<<endl;

}

ExitThread (0);

return 0;

}

DWORD WINAPI Thread1Proc(

LPVOID lpParameter

)

{

while(index++<1000)

{

cout<<"Threa 1 is running!"<<endl;

}

return 0;

}

DWORD WINAPI Thread2Proc(

LPVOID lpParameter

)

{

while(index++<1000)

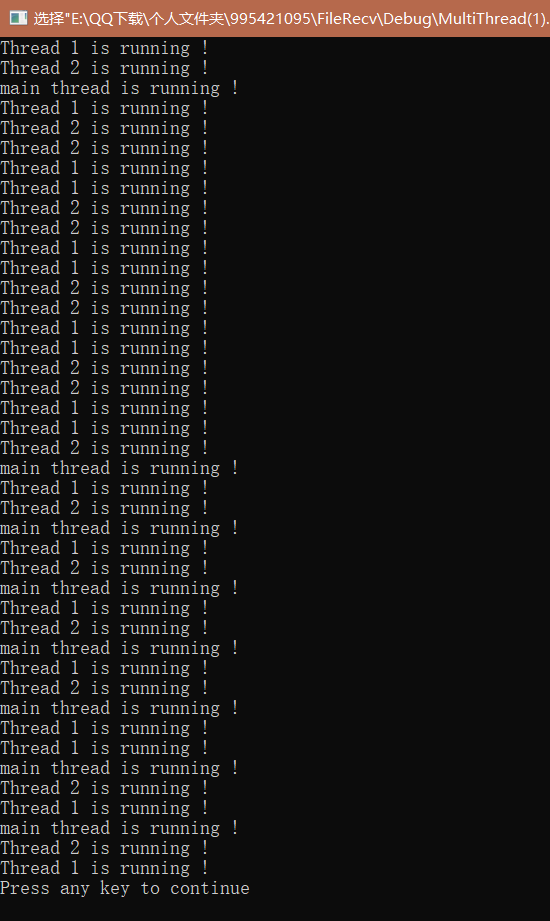
{

cout<<"Threa 2 is running!"<<endl;

}

return 0;

}



**图一**

**2.利用2个线程模拟火车票销售系统**

该程序段利用上例代码实现利用2个线程模拟火车票销售系统,其主要代码如下所示。需要注意的是该段程序是含有隐患的，注意考虑其隐患所在。

#include <windows.h>

#include <iostream.h>

Threa1Proc(

NULL,

0,

Threa2Proc=CreateThread(NULL,

0,

Threa2Proc,

NULL,

0,

NULL);

CloseHandle(hThread1);

CloseHandle(hThread2);

Sleep(5000);

ExitThread(0);

return 0;

}

DWORD WINAPI Thread1Proc(

LPVOID lpParameter

)

{

while(TRUE)

{

if(tickets>0)

{

Sleep(1);

cout<<"thread 1 sell ticket:"<<tickets--<<endl;

}

else

break;

}

return 0;

}

DWORD WINAPI Threa2Proc(

LPVOID lpParameter

)

{

while(TRUE)

{

if(tickets>0)

{

Sleep(1);

cout<<"thread 2 sell ticket:"<<tickets--<<endl;

}

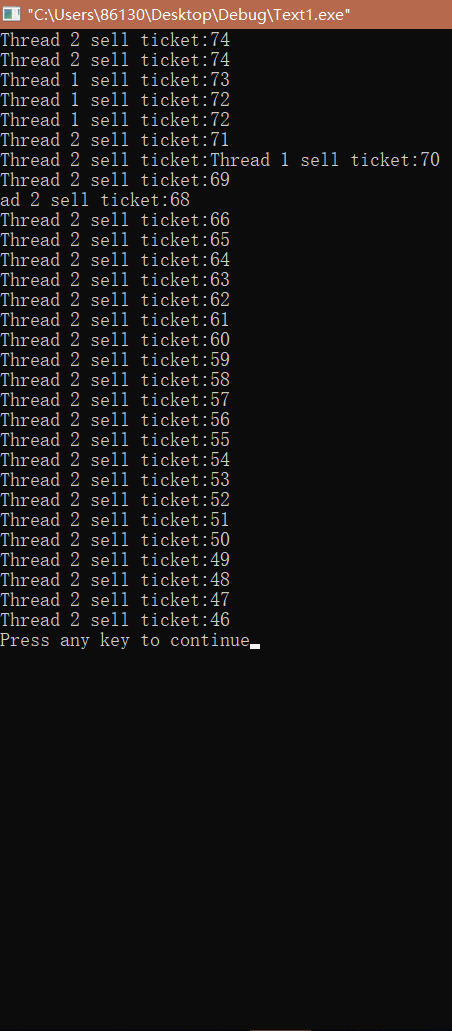
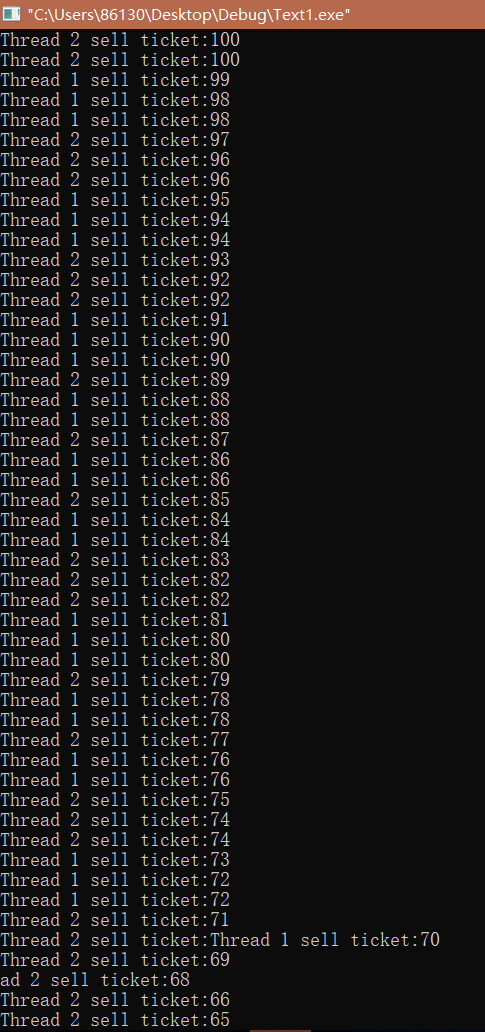
else

break;

}

return 0;

}



**图二**

**五、实验总结**

**1.创建线程:**

CreateThread()完成线程的创建，它在调用进程的地址空间上创建一个线程，执行指定的函数，并返回新建立的线程的句柄。

**2.撤销线程:**

ExitThread()函数 ，强制线程终止运行；该函数将终止线程的运行，并导致操作系统清除该线程使用的所有操作系统。 C++资源（如C++类对象）将不被撤销。

**3.终止线程:**

TerminateThread()函数也能终止线程的运行，该函数与ExitThead()区别在于，ExitThread()在撤销线程时将该线程所拥有的资源全部归还给系统，而TerminateThread()不归还资源。

**4.关闭句柄**

CloseHandle()用于关闭已打开的对象的句柄，其作用与释放动态申请的内存空间类似，这样可以释放系统资源，使线程安全运行。

由于sleep函数的介入，运行顺序被更改。我们可以通过CreateThread()创建线程，ExitThread()函数撤销线程。多线程环境下，线程之间的运行是同时进行的，通过sleep函数可以控制线程的运行时间，三个线程间的运行状态不互相影响