# 自动重置事件对象与手动重置事件对象

## 一、创建事件对象

创建事件对象的函数是CreateEvent，函数参数如下：

HANDLE CreateEvent( LPSECURITY\_ATTRIBUTES lpEventAttributes,

BOOL bManualReset,

BOOL bInitialState,

LPTSTR lpName );

函数说明请参考MSDN。

## 二、自动重置对象与手动重置对象的区别

自动重置事件对象与手动重置对象的区别是：

1、自动重置对象使用SetEvent将其设置为触发状态，这时使用WaitForSingleObject函数等待事件对象的线程会有机会得到事件对象，WaitForSingleObject函数会自动将事件对象设置为非触发状态，在执行完线程代码后，需要用SetEvent函数来将其设置为触发状态，以使其它线程获得该事件对象。

2、手动重置对象使用SetEvent将其设置为触发状态，而使用WaitForSingleObject函数的等待事件对象的线程得到事件对象，但不会将事件对象设置为非触发状态，必须使用ReSetEvent函数来将其设置为触发状态。

## 三、例程

请看以下例程：

例程一：

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

HANDLE hThreadOne;

HANDLE hThreadTwo;

HANDLE g\_hEvent;

DWORD WINAPI ThreadOneProc(LPVOID lpParam)

{

WaitForSingleObject(g\_hEvent,INFINITE);

for (int i=0;i<50;i++)

{

cout<<"ThreadOne is Running!"<<endl;

}

SetEvent(g\_hEvent);

return 0;

}

DWORD WINAPI ThreadTwoProc(LPVOID lpParam)

{

WaitForSingleObject(g\_hEvent,INFINITE);

for (int i=0;i<50;i++)

{

cout<<"ThreadTwo is Running!"<<endl;

}

SetEvent(g\_hEvent);

return 0;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

g\_hEvent=CreateEvent(NULL,FALSE,FALSE,NULL);//自动重置

cout<<"Initialzing..."<<endl;

int x=0;

DWORD ThreadOneID,ThreadTwoID;;

hThreadOne=CreateThread(NULL,0,ThreadOneProc,&x,CREATE\_SUSPENDED,&ThreadOneID);

hThreadTwo=CreateThread(NULL,0,ThreadTwoProc,&x,CREATE\_SUSPENDED,&ThreadTwoID);

ResumeThread(hThreadOne);

ResumeThread(hThreadTwo);

//initailze is over

SetEvent(g\_hEvent);

HANDLE h[2];

h[0]=hThreadOne;

h[1]=hThreadTwo;

WaitForMultipleObjects(2,h,TRUE,INFINITE);

cout<<"Both thread is over"<<endl;

system("pause");

return 0;

}

例程一中创建了自动重置对象，自动重置对象使用WaiForSingleObject函数会自动将事件对象设置为非触发状态，以使拥有该事件对象的统一线程得以独立执行，在执行完之后，需要使用SetEvent函数将事件对象设置为非触发状态，以使其它等待它的线程得到执行的机会。代码中，由于在线程代码中使用了WaitForSingleObject(g\_hEvent,INFINITE)和SetEvent(g\_hEvent)对代码进行了保护，因而其间的代码执行不会被打断，线程一和线程二是前后进行的。

例程二：

// EventDemo.cpp : 定义控制台应用程序的入口点。

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

HANDLE hThreadOne;

HANDLE hThreadTwo;

HANDLE g\_hEvent;

DWORD WINAPI ThreadOneProc(LPVOID lpParam)

{

WaitForSingleObject(g\_hEvent,INFINITE);

for (int i=0;i<50;i++)

{

cout<<"ThreadOne is Running!"<<endl;

}

return 0;

}

DWORD WINAPI ThreadTwoProc(LPVOID lpParam)

{

WaitForSingleObject(g\_hEvent,INFINITE);

for (int i=0;i<50;i++)

{

cout<<"ThreadTwo is Running!"<<endl;

}

return 0;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

g\_hEvent=CreateEvent(NULL,TRUE,FALSE,NULL);//手动重置

cout<<"Initialzing..."<<endl;

int x=0;

DWORD ThreadOneID,ThreadTwoID;;

hThreadOne=CreateThread(NULL,0,ThreadOneProc,&x,CREATE\_SUSPENDED,&ThreadOneID);

hThreadTwo=CreateThread(NULL,0,ThreadTwoProc,&x,CREATE\_SUSPENDED,&ThreadTwoID);

ResumeThread(hThreadOne);

ResumeThread(hThreadTwo);

//initailze is over

SetEvent(g\_hEvent);

HANDLE h[2];

h[0]=hThreadOne;

h[1]=hThreadTwo;

WaitForMultipleObjects(2,h,TRUE,INFINITE);

cout<<"Both thread is over"<<endl;

system("pause");

return 0;

}

例程二中创建的是手动重置对象，手动重置事件对象使用WaitForSingleObject函数来等待事件对象，但在得到事件对象之后并不会将事件对象设置为非触发态，因此如果多个线程使用WaitForSingleObject函数来等待事件对象，可以多个线程同时会得到执行的机会。在代码中，由于两个子线程都得到事件对象，因此两者是不断交替执行的。