# 主线程等待子线程结束再往下执行的方法

## 方法一（不推荐）

1、原理：使用互斥对象，在主程序中创建互斥对象，创建时，不使主线程拥有互斥对象所有权，发起线程后，使子线程拥有所有权，再执行完子线程后，再释放所有权。

2、互斥对象创建的方法：

定义全局变量：HANDLE hMutex;

在主线程中创建互斥对象：hMutex=CreateMutex(NULL,FALSE,NULL);第二个参数为FALSE，互斥对象创建后主线程不拥有所有权，当为TRUE时，主线程拥有初始的所有权，使用ReleaseMutex(hMutex)可以释放所有权，使其它线程可以有机会得到所有权。

3、创建子线程，并在子线程中获取互斥对象所有权，再执行完之后释放之。注意，在创建子线程后，要使用Sleep，使主线程睡眠一段时间，以使子线程得到执行的机会。

WaitForSingleObject(hMutex,INFINITE);

…

ReleaseMutex(hMutex);

4、主线程中等待所有权

DWORD result=WaitForSingleObject(hMutex,INFINITE);

if (result==WAIT\_TIMEOUT)

cout<<"等待超时..."<<endl;

else if (result==WAIT\_OBJECT\_0)

cout<<"子线程已结束，当前有信号!"<<endl;

else

cout<<"句柄无效!"<<endl;

cout<<"主程序结束！"<<endl;

5、代码

#include "stdafx.h"

#include <iostream>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

HANDLE hThread;

HANDLE hMutex;

DWORD WINAPI FuncProc(LPVOID lpParmeter)

{

WaitForSingleObject(hMutex,INFINITE);

cout<<"子程序开始！"<<endl;

for (int i=0;i<10;i++)

{

cout<<"子程序执行中..."<<endl;

Sleep(500);

}

cout<<"子程序结束！"<<endl;

ReleaseMutex(hMutex);

return 0;

}

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

{

cout<<"主程序开始！"<<endl;

hMutex=CreateMutex(NULL,FALSE,NULL);

hThread=CreateThread(NULL,0,FuncProc,NULL,0,NULL);

CloseHandle(hThread);

Sleep(1000);

DWORD result=WaitForSingleObject(hMutex,INFINITE);

if (result==WAIT\_TIMEOUT)

cout<<"等待超时..."<<endl;

else if (result==WAIT\_OBJECT\_0)

cout<<"子线程已结束，当前有信号!"<<endl;

else

cout<<"句柄无效!"<<endl;

cout<<"主程序结束！"<<endl;

system("pause");

return 0;

}

## 方法二（推荐）

1、原理，使用内核对象。进程、线程内核对象内部有一个布尔变量，当系统创建内核对象的时候会把这个变量的值初始化为FALSE（未触发），当进程终止的时候，操作系统会把相应的内核对象中的这个布尔值设为TRUE，表示该对象已被触发。如果父线程需要等待一个线程结束，那么我们可以编写代码让其直接进入睡眠民，直到标识子线程的内核对象被触发为止。

2、实现：

直接使用WaitForSingleObject(hThread,INFINITE)即可。

3、代码：

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

DWORD WINAPI SubThreadProc(PVOID pParam)

{

for (int i=0;i<100;i++)

Sleep(1);

cout<<"SubThread is Over!"<<endl;

return 0;

}

HANDLE hThread;

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

{

int x=0;

DWORD dwThreadID;

hThread=CreateThread(NULL,0,SubThreadProc,&x,0,&dwThreadID);

WaitForSingleObject(hThread,INFINITE);

CloseHandle(hThread);

cout<<"Main Thread is over!"<<endl;

system("pause");

return 0;

}