MFC Multithreading and Timer()

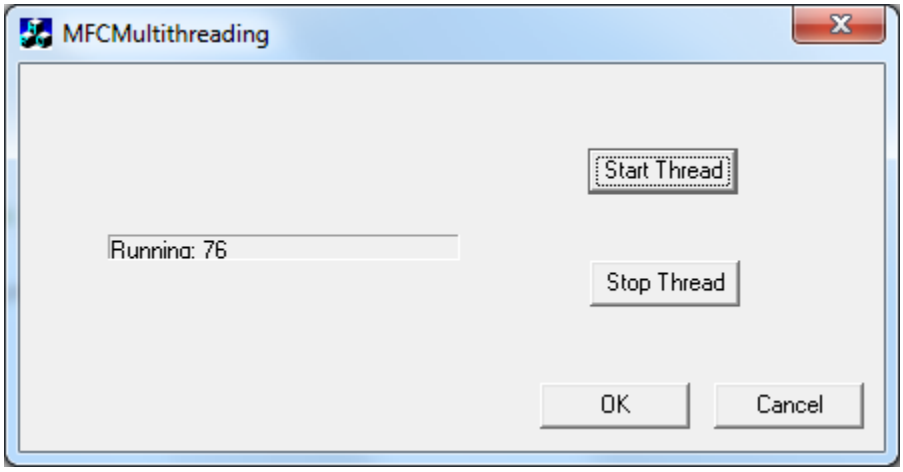
10-27-2020 William Xu

摘要:

本文描写如何使用 MFC 的 Multithreading 功能以及定时器 Timer()在 Multithreading 中的应用. 在附录中(Appendix B)还摘取了专家的一些建议: Multithreading 是一种看似很花哨但是不实用的技巧. 虽然有各种同步的方式, 但在多资源的共享方面, Multithreading 是产生 Bug 的源头.

I. MFC Sample Program MFCMultithreading

MFCMultithreading 是一个很简单的 VC++ 6.0 MFC 应用程序. 从下图中可以看到它有两个按钮, "Start Thread" 和 " Stop Thread". 启动"Start Thread"按钮, 建立新的 thread, 左边的计数器开始工作, 显示 "Running XX…". 启动"Stop Thread"按钮, thread 结束,计数器停止计数.



按照 MFC 原理, **MFC** 程序本身就是第 1 个 **thread.** 所以启动 "Start Thread"按钮是为它建立了第 2 个 **thread.** 故可称为 Multithreading.

下面描述新的 **thread** 如何启动 / 停止.

1) **Function AfxBeginThread (par1, par2)** 启动 **thread.**

Par1 - The address of the controlling function

Par2 - The parameter to be passed to the controlling function

在这里参数 1 就是 thread 要执行的程序；参数 2 就是要送到执行程序中的变量. 在我们的程序中这一条命令写为:

AfxBeginThread(MyThreadProc, 0);

```
void CMFCMultithreadingDlg::OnBnClickedButtonStart() {
  currValue = 0;
  maxValue = 5000;
  stopNow = 0;
  m_ctrlStatus.SetWindowText("Starting...");
  SetTimer(1234, 333, 0); // 3 times per second
  AfxBeginThread ( MyThreadProc, 0 ); // <<== START THE THREAD
}
```

其中, MyThreadProc()是一个独立的程序, 在本程序中它只是计数而已.

```
//
//Sample function for using in AfxBeginThread
//
UINT MyThreadProc ( LPVOID Param )
{
  while (!stopNow && (currValue < maxValue)) {
    currValue++;
    Sleep(50); // would do some work here
  }
  return TRUE;
}
```

2) 在本程序中, 停止 **thread** 的功能用停止执行 MyThreadProc()程序来进行. 在启动 **thread** 的命令中定义了 **thread** 的执行程序是 MyThreadProc(), 所以只要它停止, 整个 **thread** 就结束了.
注意: 在这里 **stopNow** 是个全局变量, 当它变成 **TRUE,** MyThreadProc()就停止了.

```
void CMFCMultithreadingDlg::OnBnClickedButtonStop() {
  stopNow = TRUE;
  KillTimer(1234);
  m_ctrlStatus.SetWindowText("Stopped");
}
```

II. 定时器 Timer()在 Multithreading 中的应用

下面描述如何启动 / 停止 MFC Timer()

1) 在 CMFCMultithreadingDlg.h Class H 文件中说明

```
class CMFCMultithreadingDlg : public CDialog
{
// Construction
… …
public:

// Implementation
protected:
    HICON m_hIcon;
    void OnTimer(UINT_PTR nIDEvent);
… …
};
```

2) 在 CMFCMultithreadingDlg.cpp 文件中说明 1

```
BEGIN_MESSAGE_MAP(CMFCMultithreadingDlg, CDialog)
    //{{AFX_MSG_MAP(CMFCMultithreadingDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_TIMER()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BUTTON_START, OnBnClickedButtonStart)
    ON_BN_CLICKED(IDC_BUTTON_STOP, OnBnClickedButtonStop)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

3) 在 CMFCMultithreadingDlg.cpp 文件中说明 2

```
void CMFCMultithreadingDlg::OnTimer(UINT_PTR nIDEvent) {
  // TODO: Add your message handler code here and/or call default
  CString sStatusMsg;
  sStatusMsg.Format("Running: %d", currValue);
  m_ctrlStatus.SetWindowText(sStatusMsg);
  CDialog::OnTimer(nIDEvent);
}
```

4) 启动

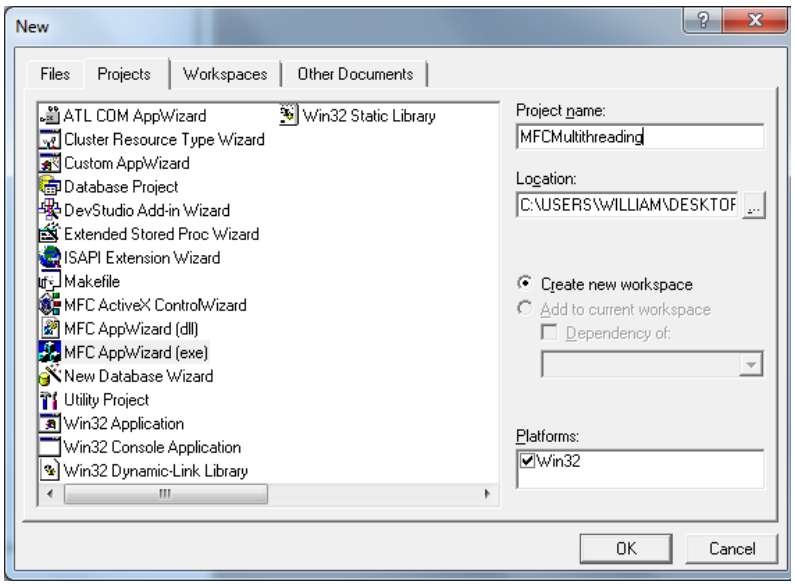
```
void CMFCMultithreadingDlg::OnBnClickedButtonStart() {
  // TODO: Add your control notification handler code here
  currValue = 0;
  maxValue = 5000;
  stopNow = 0;
  m_ctrlStatus.SetWindowText("Starting...");

  SetTimer(1234, 333, 0); // 3 times per second

  AfxBeginThread(MyThreadProc, 0); // <<== START THE THREAD
}
```

5) 停止

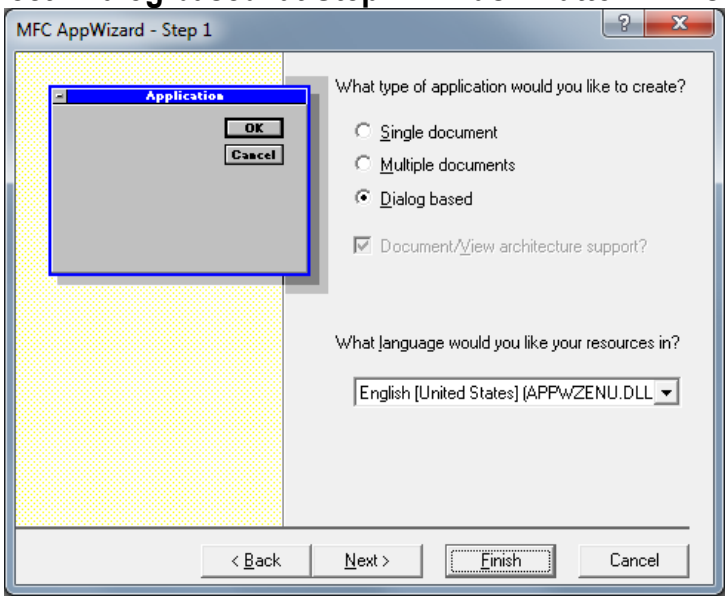
```
void CMFCMultithreadingDlg::OnBnClickedButtonStop() {
  // TODO: Add your control notification handler code here
  stopNow = TRUE;
  KillTimer(1234);
  m_ctrlStatus.SetWindowText("Stopped");
}
```

III. 如何建立 MFCMultithreading Program

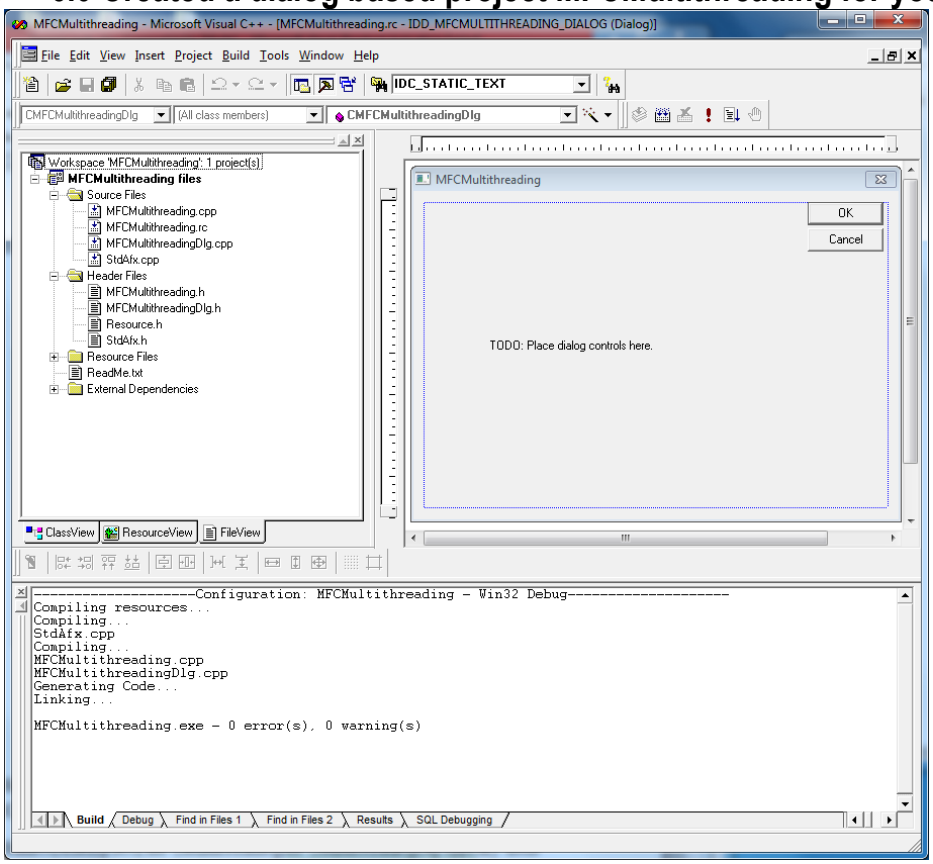
1. **VC++ 6.0. File-> New-> Projects-> MFCAppWizard(exe) -> OK**
   Be sure to set the project name as 'MFCMultithreading'.



2. **Select ‘Dialog based’ at Step 1 -> Push Button ‘Finish’**



3. **VC++ 6.0 Created a dialog based project MFCMultithreading for you**

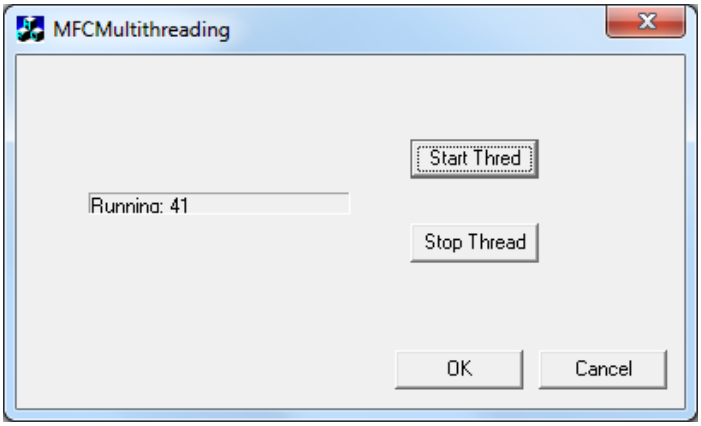


4. **Replace MFCMultithreadingDlg.cpp and MFCMultithreadingDlg.h two files with the same name files listed at the Appendix A.**

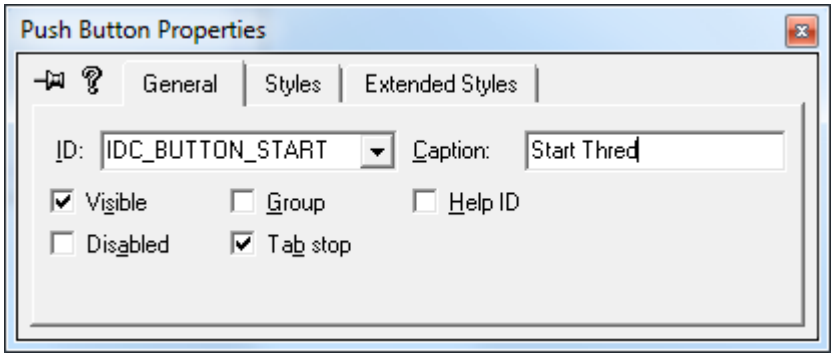
5. **Hit F7 to build the project. You will see three errors**:

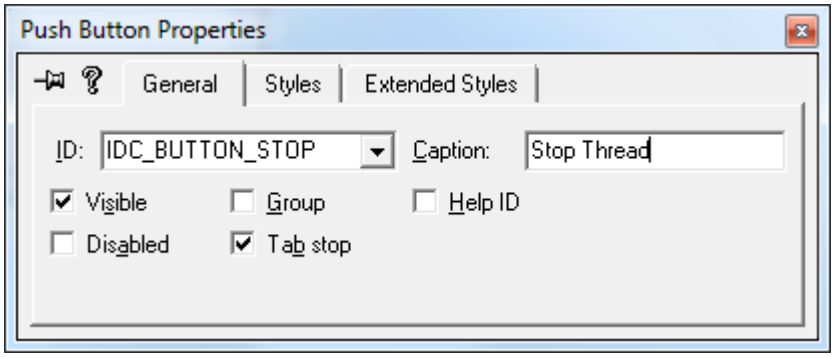
MFCMultithreadingDlg.cpp(79) : error C2065: 'IDC_STATIC_TEXT' : undeclared identifier
MFCMultithreadingDlg.cpp(92) : error C2065: 'IDC_BUTTON_START' : undeclared identifier
MFCMultithreadingDlg.cpp(93) : error C2065: 'IDC_BUTTON_STOP' : undeclared identifier

6. **Add two buttons, "Start Thread" and "Stop Thread".**

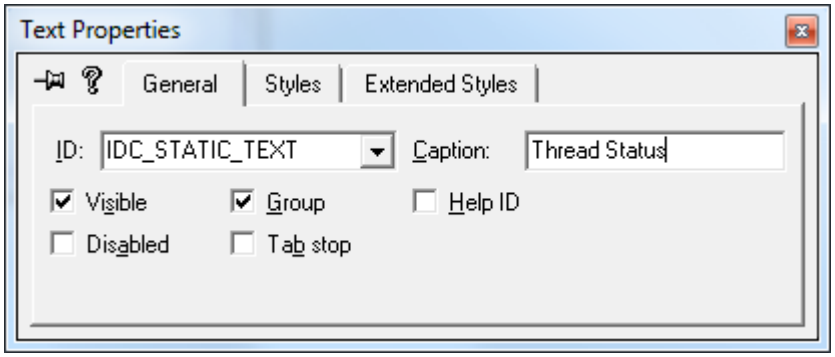


7. **Set both Buttons Properties as below**





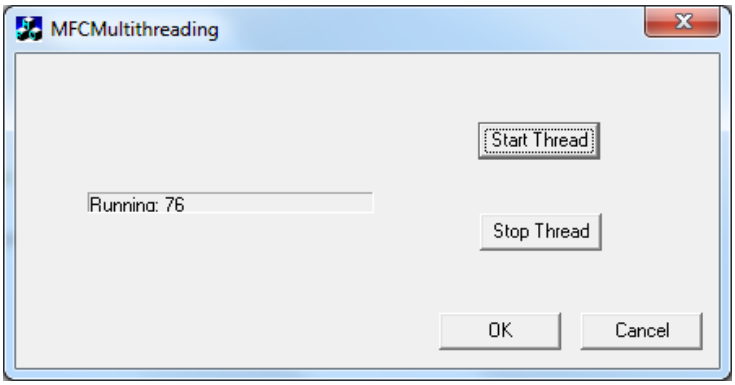
8. **Replace existed static text's properties as below. This text area will be used as the new thread status field.**



9. **Hit F7 again to re-build the project. It should be OK now. You will see:**

```
--------------------Configuration: MFCMultithreading - Win32 Debug--------------------
Compiling resources...
Compiling...
StdAfx.cpp
Compiling...
MFCMultithreading.cpp
MFCMultithreadingDlg.cpp
Generating Code...
Linking...

MFCMultithreading.exe - 0 error(s), 0 warning(s)
```

10. **Hit CTRL/F5 to run the program.**



Appendix A Source code

(MFCMultithreadingDlg.cpp & MFCMultithreadingDlg.h)

```
//
// MFCMultithreadingDlg.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "MFCMultithreading.h"
#include "MFCMultithreadingDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

int currValue;
int maxValue;
BOOL stopNow;

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
```

```
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMFCMultithreadingDlg dialog

CMFCMultithreadingDlg::CMFCMultithreadingDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMFCMultithreadingDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMFCMultithreadingDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMFCMultithreadingDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMFCMultithreadingDlg)
	DDX_Control(pDX, IDC_STATIC_TEXT, m_ctrlStatus);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMFCMultithreadingDlg, CDialog)
	//{{AFX_MSG_MAP(CMFCMultithreadingDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()

	// It is important to add ON_WM_TIMER() here for using in OnTimer(UINT_PTR nIDEvent)
	ON_WM_TIMER()

	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON_START, OnBnClickedButtonStart)
	ON_BN_CLICKED(IDC_BUTTON_STOP, OnBnClickedButtonStop)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMFCMultithreadingDlg message handlers
```

```
BOOL CMFCMultithreadingDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMFCMultithreadingDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
```

```
void CMFCMultithreadingDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CMFCMultithreadingDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}


UINT MyThreadProc(LPVOID Param) //Sample function for using in AfxBeginThread
{
  while (!stopNow && (currValue < maxValue)) {
    currValue++;
    Sleep(50); // would do some work here
  }
  return TRUE;
}



void CMFCMultithreadingDlg::OnTimer(UINT_PTR nIDEvent) {
  // TODO: Add your message handler code here and/or call default
  CString sStatusMsg;
```

```
  sStatusMsg.Format("Running: %d", currValue);
  m_ctrlStatus.SetWindowText(sStatusMsg);

  CDialog::OnTimer(nIDEvent);
}


void CMFCMultithreadingDlg::OnBnClickedButtonStart() {
  // TODO: Add your control notification handler code here
  currValue = 0;
  maxValue = 5000;
  stopNow = 0;
  m_ctrlStatus.SetWindowText("Starting...");

  SetTimer(1234, 333, 0); // 3 times per second

  AfxBeginThread(MyThreadProc, 0); // <<== START THE THREAD
}

void CMFCMultithreadingDlg::OnBnClickedButtonStop() {
  // TODO: Add your control notification handler code here
  stopNow = TRUE;
  KillTimer(1234);
  m_ctrlStatus.SetWindowText("Stopped");
}


//
// MFCMultithreadingDlg.h : header file
//

#if !defined(AFX_MFCMULTITHREADINGDLG_H__E27E077C_6C55_496C_A4A9_05BD787E1FB6__INCLU
DED_)
#define
AFX_MFCMULTITHREADINGDLG_H__E27E077C_6C55_496C_A4A9_05BD787E1FB6__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CMFCMultithreadingDlg dialog

class CMFCMultithreadingDlg : public CDialog
{
// Construction
```

```
public:
	CMFCMultithreadingDlg(CWnd* pParent = NULL);	// standard constructor

// Dialog Data
	//{{AFX_DATA(CMFCMultithreadingDlg)
	enum { IDD = IDD_MFCMULTITHREADING_DIALOG };
	CStatic  m_ctrlStatus;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMFCMultithreadingDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;
	void OnTimer(UINT_PTR nIDEvent);

	// Generated message map functions
	//{{AFX_MSG(CMFCMultithreadingDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnBnClickedButtonStart();
	afx_msg void OnBnClickedButtonStop();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif
// !defined(AFX_MFCMULTITHREADINGDLG_H__E27E077C_6C55_496C_A4A9_05BD787E1FB6__INCLU
DED_)
```

Appendix B

专家的一些建议：Multithreading 是一种看似很花哨但是不实用的技巧. 虽然有各种同步的方式, 但在多资源的共享方面, Multithreading 是产生 Bug 的源头.

Book: Professional MFC Visual C++ 5, Mike Blaszczak, P526. 1997.

Applications for Multithreading

If you think about it for a bit, you might wonder what the point is of having multiple threads. After all, it's not like two things are really happening at once, and going through these hoops to let the operating system pretend that two things can happen at once is more trouble than it's worth, isn't it?

如果稍微考虑一下，您可能会想知道运用多个 **thread** 的意义是什么。毕竟，这并不是真的同时发生了两件事，而经过这些处理让操作系统假装一次可以发生两件事. 实际上这并不是一件值得的事，不是吗？

Largely, you're right - it is more trouble than it's worth. There's usually no real reason to write a multithreaded application. Some marketing guys hear that 'multithreaded' is a cool buzzword and harass the developers to implement many threads, like some nightmarish Dilbert cartoon, or some developer gets it into their head that they won't be cool unless they use multiple threads. So, in order to look good in front of their friends, they scamper off and write an application that creates threads that create threads to create threads.

总的来说，您是对的 -- 实际上这里麻烦事多于其所值。通常没有真正的理由来编写多 **thread** 应用程序。一些营销人员听说〝多 **thread** 程序〞是一个很时髦的流行词，并且鼓动开发人员实现许多多 **thread** 程序，例如一些噩梦般的迪尔伯特（**Dilbert**）卡通漫画，或者某些开发人员意识到，除非使用多个 **thread** 程序，否则它们不会很酷。因此，为了在朋友面前看起来不错，他们分散了精力，编写了一个个多 **thread** 的应用程序，他们实际上是为创建多 **thread** 程序而创建多 **thread** 程序。

In fact, what you get are performance bottlenecks. The operating system has to take some amount of time to switch from thread to thread - you can't get away from that. There are also a few things to worry about when you're trying to communicate between threads - more on that later. So, unless you absolutely need the threads, there's really no point.

You're slowing down your application and making work for yourself when you should be out watching ice hockey games or playing with your motorcycle.

实际上，**(**用多 **thread)**您将碰到性能瓶颈。操作系统必须花费一些时间从一个 **thread** 切换到另一个 **thread --** 您无法摆脱这一点。当您尝试在多个 **thread** 之间进行通信时，还需要担心一些事情 **--** 稍后再讨论。因此，除非您绝对需要多 **thread**，否则实际上没有任何意义。就像当您出门看冰球比赛或骑摩托车时，您会放慢自己动手做程序的速度**.**