# 学习大纲

|  |
| --- |
|  |

## AfxBeginThread函数的用法

|  |
| --- |
| 调用此函数以创建新线程。  C++复制  CWinThread\* AfxBeginThread(  AFX\_THREADPROC pfnThreadProc,  LPVOID pParam,  int nPriority = THREAD\_PRIORITY\_NORMAL,  UINT nStackSize = 0,  DWORD dwCreateFlags = 0,  LPSECURITY\_ATTRIBUTES lpSecurityAttrs = NULL);  CWinThread\* AfxBeginThread(  CRuntimeClass\* pThreadClass,  int nPriority = THREAD\_PRIORITY\_NORMAL,  UINT nStackSize = 0,  DWORD dwCreateFlags = 0,  LPSECURITY\_ATTRIBUTES lpSecurityAttrs = NULL); 参数 *pfnThreadProc* 指向工作线程的控制函数。 指针不能是 NULL。 此函数必须声明如下：  UINT \_\_cdecl MyControllingFunction( LPVOID pParam );  *pThreadClass* 派生自 [CWinThread](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/cwinthread-class?view=msvc-170) 的对象的 RUNTIME\_CLASS。  *pParam* 要传递给控制函数的参数。  *nPriority* 要为线程设置的优先级。 有关可用优先级的完整列表和说明，请参阅 Windows SDK 中的 [SetThreadPriority](https://learn.microsoft.com/zh-cn/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadpriority)。  *nStackSize* 指定新线程堆栈的大小（以字节为单位）。 如果为 0，则堆栈大小默认为与所创建线程相同的堆栈大小。  *dwCreateFlags* 指定一个用于控制线程创建的额外标志。 此标志可以包含两个值之一：   * CREATE\_SUSPENDED 以 1 的挂起计数启动线程。 如果要在线程开始运行之前初始化 CWinThread 对象的任何成员数据（例如 [m\_bAutoDelete](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/cwinthread-class?view=msvc-170#m_bautodelete) 或派生类的任何成员），请使用 CREATE\_SUSPENDED。 初始化完成后，使用 [CWinThread::ResumeThread](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/cwinthread-class?view=msvc-170#resumethread) 启动线程运行。 在调用 CWinThread::ResumeThread 之前，线程不会执行。 * **0** 创建后立即启动线程。   *lpSecurityAttrs* 指向用于指定线程安全属性的 [SECURITY\_ATTRIBUTES](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/aa379560(v=vs.85)) 结构。 如果为 NULL，则使用与所创建线程相同的安全属性。 有关此结构的详细信息，请参阅 Windows SDK。 返回值 指向新创建的线程对象的指针，如果发生故障，则为 NULL。 备注 AfxBeginThread 的第一个窗体创建一个工作线程。 第二个窗体创建一个可用作用户界面线程或工作线程的线程。  AfxBeginThread 创建一个新的 CWinThread 对象，调用其 [CreateThread](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/cwinthread-class?view=msvc-170#createthread) 函数以开始执行线程，并返回指向该线程的指针。 在整个过程中都会进行检查，以确保在创建的任何部分失败时正确解除分配所有对象。 若要结束线程，请从线程内部调用 [AfxEndThread](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/application-information-and-management?view=msvc-170#afxendthread)，或从工作线程的控制函数返回。  应用程序必须启用多线程；否则，此函数将失败。 有关启用多线程处理的详细信息，请参阅 [/MD、/MT、/LD（使用运行时库）](https://learn.microsoft.com/zh-cn/cpp/build/reference/md-mt-ld-use-run-time-library?view=msvc-170)。  有关 AfxBeginThread 的详细信息，请参阅文章[多线程处理：创建工作线程](https://learn.microsoft.com/zh-cn/cpp/parallel/multithreading-creating-worker-threads?view=msvc-170)和[多线程处理：创建用户界面线程](https://learn.microsoft.com/zh-cn/cpp/parallel/multithreading-creating-user-interface-threads?view=msvc-170)。 示例 请参阅 [CSocket::Attach](https://learn.microsoft.com/zh-cn/cpp/mfc/reference/csocket-class?view=msvc-170#attach) 的示例。 要求  afxwin.h |

### 注意：MFC的线程函数和win32有点不一样，他是有返回值的

## 在多线程程序中，业务逻辑代码一般有其他线程来做。主线程只是负责界面响应，这样子程序就不会方式阻塞

# 演练

## 1.新建一个文件夹Lesson54-MFC-Create-Thread然后在里面创建一个FMC基于对话框的应用程序，取名：MFCThread

|  |
| --- |
|  |

## 2.把对话框上面的取消按钮删除，然后把确定按钮的文本改为退出，双击进入点击事件处理函数，添加下面的代码

|  |
| --- |
|  |
|  |

## 3.我们给对话框添加3个按钮，然后在按钮上面添加3个静态文本作为输出。

|  |
| --- |
|  |

## 4.为了方便比较，我们在第一个按钮的点击事件中使用主线程来响应，在第二个按钮的点击事件中创建一个线程来响应，第一个按钮的响应代码如下

|  |
| --- |
|  |

### 当我们按下按钮1，程序开始工作，但是此时你尝试点击对话框的任何部分，你会发现对话框没有响应。

## 5.然后，我们在按钮2的点击事件中创建一个线程来做这个事情，注意：在MFC中，线程函数必须是全局函数不能是类的成员函数，我们先在头文件里面声明，然后在cpp文件里面定义。

|  |
| --- |
|  |
|  |

## 6.在按钮2的点击事件里面创建线程

|  |
| --- |
|  |

## 7,我们在按钮3的点击事件响应函数里面也创建一个线程

|  |
| --- |
|  |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 注意：多线程是window程序设计里面非常重要的方法，需要好好掌握

## 这一节的学习到此为止，完整代码如下

## MFCThreadDlg.h

|  |
| --- |
| // MFCThreadDlg.h : 头文件  //  #pragma once  #include "afxwin.h"  UINT ThreadProc(LPVOID lpvoid);//线程函数必须是全局函数，在类外面声明  UINT ThreadProc2(LPVOID lpvoid);//线程函数必须是全局函数，在类外面声明  // CMFCThreadDlg 对话框  class CMFCThreadDlg : public CDialogEx  {  // 构造  public:  CMFCThreadDlg(CWnd\* pParent = NULL); // 标准构造函数  // 对话框数据  enum { IDD = IDD\_MFCTHREAD\_DIALOG };  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持  // 实现  protected:  HICON m\_hIcon;  // 生成的消息映射函数  virtual BOOL OnInitDialog();  afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);  afx\_msg void OnPaint();  afx\_msg HCURSOR OnQueryDragIcon();  DECLARE\_MESSAGE\_MAP()  public:  afx\_msg void OnBnClickedOk();  afx\_msg void OnBnClickedButton1();  afx\_msg void OnBnClickedButton2();  afx\_msg void OnBnClickedButton3();  }; |

## MFCThreadDlg.cpp

|  |
| --- |
| // MFCThreadDlg.cpp : 实现文件  //  #include "stdafx.h"  #include "MFCThread.h"  #include "MFCThreadDlg.h"  #include "afxdialogex.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  //线程函数的定义，注意不是类的成员函数  UINT ThreadProc(LPVOID lpvoid)  {    for(int i=1;i<100;i++)  {  //这里不是在类里面，只能够使用全局函数而不能使用类的成员函数  ::SetDlgItemInt(AfxGetApp()->m\_pMainWnd->m\_hWnd,IDC\_RESULT2,i,FALSE);  Sleep(1000);  }  return 0;  }  UINT ThreadProc2(LPVOID lpvoid)  {    for(int i=1;i<100;i++)  {  //这里不是在类里面，只能够使用全局函数而不能使用类的成员函数  ::SetDlgItemInt(AfxGetApp()->m\_pMainWnd->m\_hWnd,IDC\_RESULT3,i,FALSE);  Sleep(400);  }  return 0;  }  // 用于应用程序“关于”菜单项的 CAboutDlg 对话框  class CAboutDlg : public CDialogEx  {  public:  CAboutDlg();  // 对话框数据  enum { IDD = IDD\_ABOUTBOX };  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持  // 实现  protected:  DECLARE\_MESSAGE\_MAP()  };  CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)  {  }  void CAboutDlg::DoDataExchange(CDataExchange\* pDX)  {  CDialogEx::DoDataExchange(pDX);  }  BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)  END\_MESSAGE\_MAP()  // CMFCThreadDlg 对话框  CMFCThreadDlg::CMFCThreadDlg(CWnd\* pParent /\*=NULL\*/)  : CDialogEx(CMFCThreadDlg::IDD, pParent)  {  m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);  }  void CMFCThreadDlg::DoDataExchange(CDataExchange\* pDX)  {  CDialogEx::DoDataExchange(pDX);    }  BEGIN\_MESSAGE\_MAP(CMFCThreadDlg, CDialogEx)  ON\_WM\_SYSCOMMAND()  ON\_WM\_PAINT()  ON\_WM\_QUERYDRAGICON()  ON\_BN\_CLICKED(IDOK, &CMFCThreadDlg::OnBnClickedOk)  ON\_BN\_CLICKED(IDC\_BUTTON1, &CMFCThreadDlg::OnBnClickedButton1)  ON\_BN\_CLICKED(IDC\_BUTTON2, &CMFCThreadDlg::OnBnClickedButton2)  ON\_BN\_CLICKED(IDC\_BUTTON3, &CMFCThreadDlg::OnBnClickedButton3)  END\_MESSAGE\_MAP()  // CMFCThreadDlg 消息处理程序  BOOL CMFCThreadDlg::OnInitDialog()  {  CDialogEx::OnInitDialog();  // 将“关于...”菜单项添加到系统菜单中。  // IDM\_ABOUTBOX 必须在系统命令范围内。  ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);  ASSERT(IDM\_ABOUTBOX < 0xF000);  CMenu\* pSysMenu = GetSystemMenu(FALSE);  if (pSysMenu != NULL)  {  BOOL bNameValid;  CString strAboutMenu;  bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);  ASSERT(bNameValid);  if (!strAboutMenu.IsEmpty())  {  pSysMenu->AppendMenu(MF\_SEPARATOR);  pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);  }  }  // 设置此对话框的图标。当应用程序主窗口不是对话框时，框架将自动  // 执行此操作  SetIcon(m\_hIcon, TRUE); // 设置大图标  SetIcon(m\_hIcon, FALSE); // 设置小图标  // TODO: 在此添加额外的初始化代码  return TRUE; // 除非将焦点设置到控件，否则返回 TRUE  }  void CMFCThreadDlg::OnSysCommand(UINT nID, LPARAM lParam)  {  if ((nID & 0xFFF0) == IDM\_ABOUTBOX)  {  CAboutDlg dlgAbout;  dlgAbout.DoModal();  }  else  {  CDialogEx::OnSysCommand(nID, lParam);  }  }  // 如果向对话框添加最小化按钮，则需要下面的代码  // 来绘制该图标。对于使用文档/视图模型的 MFC 应用程序，  // 这将由框架自动完成。  void CMFCThreadDlg::OnPaint()  {  if (IsIconic())  {  CPaintDC dc(this); // 用于绘制的设备上下文  SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);  // 使图标在工作区矩形中居中  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;  // 绘制图标  dc.DrawIcon(x, y, m\_hIcon);  }  else  {  CDialogEx::OnPaint();  }  }  //当用户拖动最小化窗口时系统调用此函数取得光标  //显示。  HCURSOR CMFCThreadDlg::OnQueryDragIcon()  {  return static\_cast<HCURSOR>(m\_hIcon);  }  void CMFCThreadDlg::OnBnClickedOk()//防止按下回车，程序退出  {  // TODO: 在此添加控件通知处理程序代码  if(IDOK == ::MessageBox(this->m\_hWnd,\_T("退出程序?"),\_T("退出确认"),MB\_OKCANCEL))  {  CDialogEx::OnOK();  }  }  void CMFCThreadDlg::OnBnClickedButton1()  {  // TODO: 在此添加控件通知处理程序代码  for(int i=1;i<100;i++)  {  SetDlgItemInt(IDC\_RESULT1,i+1);  Sleep(1000);  }  }  void CMFCThreadDlg::OnBnClickedButton2()  {  // TODO: 在此添加控件通知处理程序代码  AfxBeginThread(ThreadProc,NULL);//mfc中创建线程  }  void CMFCThreadDlg::OnBnClickedButton3()  {  // TODO: 在此添加控件通知处理程序代码  AfxBeginThread(ThreadProc2,NULL);//mfc中创建线程  } |