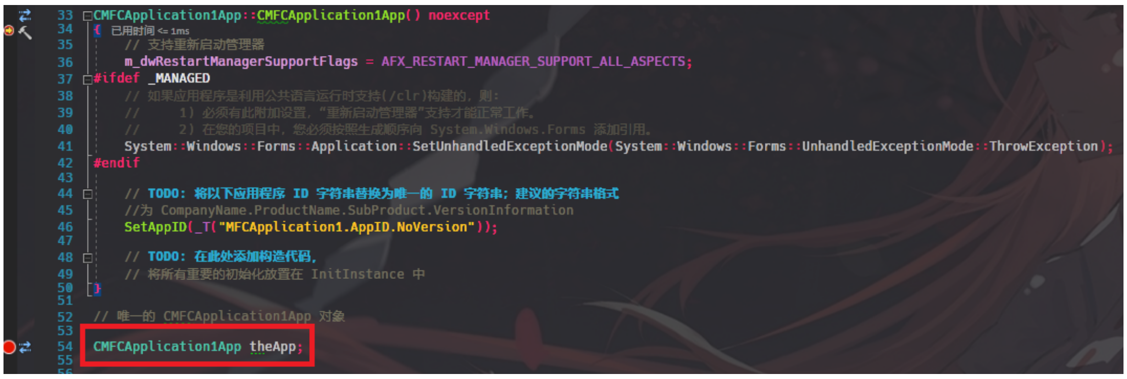
MFC入口分析

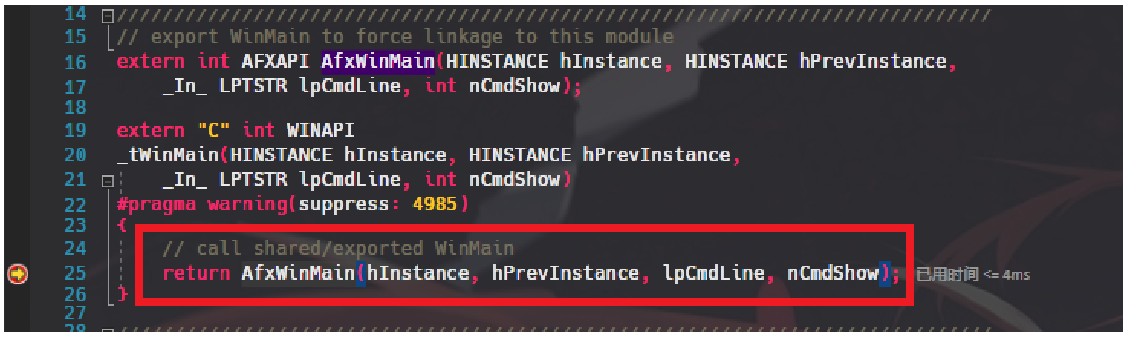
单文档

1. 先调用全局对象`theApp`的构造函数，执行全局对象的构造



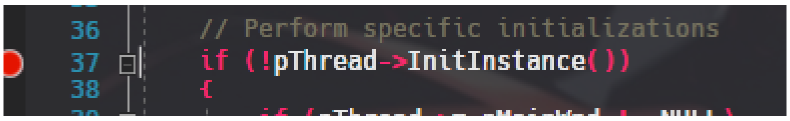
```cpp
CMFCApplication1App::CMFCApplication1App() noexcept
{ 已用时间 <= 1ms
    // 支持重新启动管理器
    m_dwRestartManagerSupportFlags = AFX_RESTART_MANAGER_SUPPORT_ALL_ASPECTS;
#ifdef _MANAGED
    // 如果应用程序是利用公共语言运行时支持(/clr)构建的，则:
    //     1) 必须有此附加设置，"重新启动管理器"支持才能正常工作。
    //     2) 在您的项目中，您必须按照生成顺序向 System.Windows.Forms 添加引用。
    System::Windows::Forms::Application::SetUnhandledExceptionMode(System::Windows::Forms::UnhandledExceptionMode::ThrowException);
#endif

    // TODO: 将以下应用程序 ID 字符串替换为唯一的 ID 字符串; 建议的字符串格式
    //为 CompanyName.ProductName.SubProduct.VersionInformation
    SetAppID(_T("MFCApplication1.AppID.NoVersion"));

    // TODO: 在此处添加构造代码,
    // 将所有重要的初始化放置在 InitInstance 中
}

// 唯一的 CMFCApplication1App 对象

CMFCApplication1App theApp;
```

2. 进入`_tWinMain`函数，调用了`AfxWinMain`

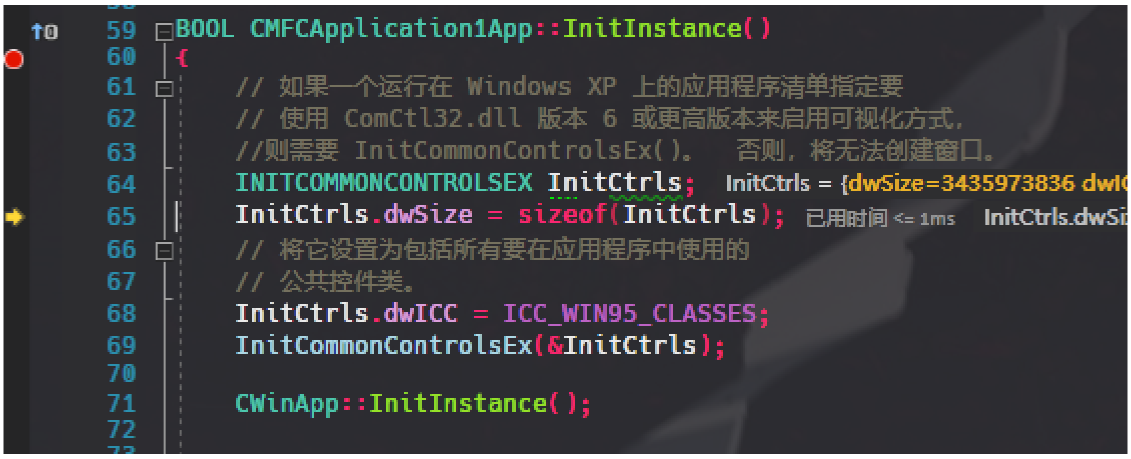


```cpp
/////////////////////////////////////////////////////////////////////////////
// export WinMain to force linkage to this module
extern int AFXAPI AfxWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    _In_ LPTSTR lpCmdLine, int nCmdShow);

extern "C" int WINAPI
_tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    _In_ LPTSTR lpCmdLine, int nCmdShow)
#pragma warning(suppress: 4985)
{
    // call shared/exported WinMain
    return AfxWinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow); 已用时间 <= 4ms
}
```

3. 在`AfxWinMain`函数中，调用了`pThread->InitInstance()`即`theApp->InitInstance()`

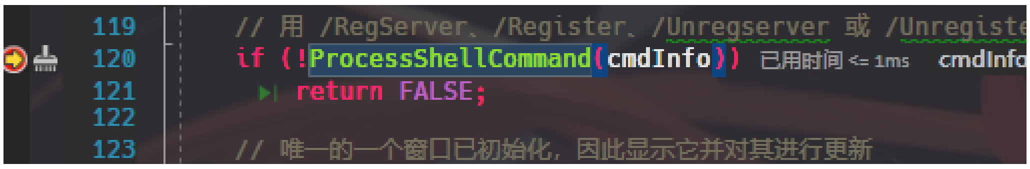


```cpp
    // Perform specific initializations
    if (!pThread->InitInstance())
    {
```

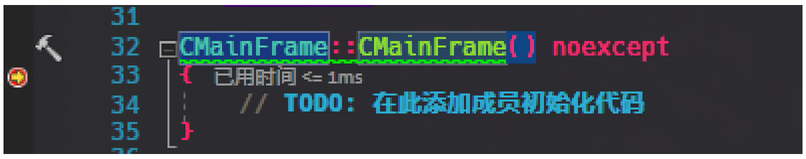


```cpp
BOOL CMFCApplication1App::InitInstance()
{
    // 如果一个运行在 Windows XP 上的应用程序清单指定要
    // 使用 ComCtl32.dll 版本 6 或更高版本来启用可视化方式,
    //则需要 InitCommonControlsEx()。  否则，将无法创建窗口。
    INITCOMMONCONTROLSEX InitCtrls;
    InitCtrls.dwSize = sizeof(InitCtrls); 已用时间 <= 1ms
    // 将它设置为包括所有要在应用程序中使用的
    // 公共控件类。
    InitCtrls.dwICC = ICC_WIN95_CLASSES;
    InitCommonControlsEx(&InitCtrls);

    CWinApp::InitInstance();
```

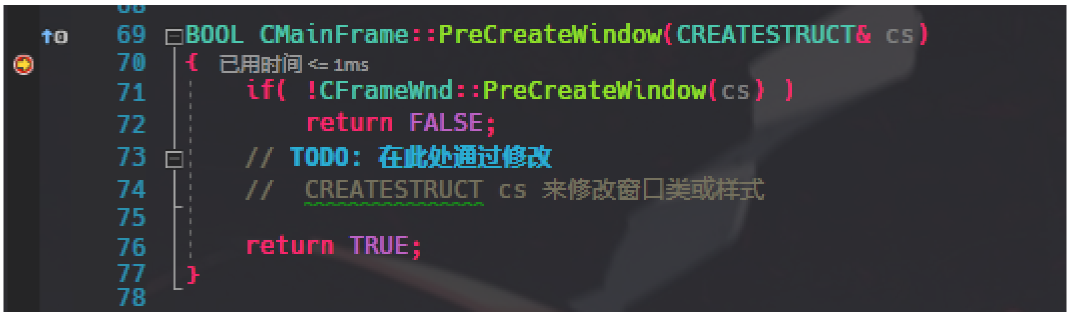
4. 在`theApp->InitInstance()`中调用了`ProcessShellCommand`函数，并在其中调用了`CMainFrame`对象的构造函数，之后又调用了`CMainFrame::PreCreateWindow`函数，又在其中调用父类的`PreCreateWindow`函数



```
    // 用 /RegServer、/Register、/Unregserver 或 /Unregiste
    if (!ProcessShellCommand(cmdInfo))  已用时间 <= 1ms   cmdInfo
        return FALSE;

    // 唯一的一个窗口已初始化，因此显示它并对其进行更新
```

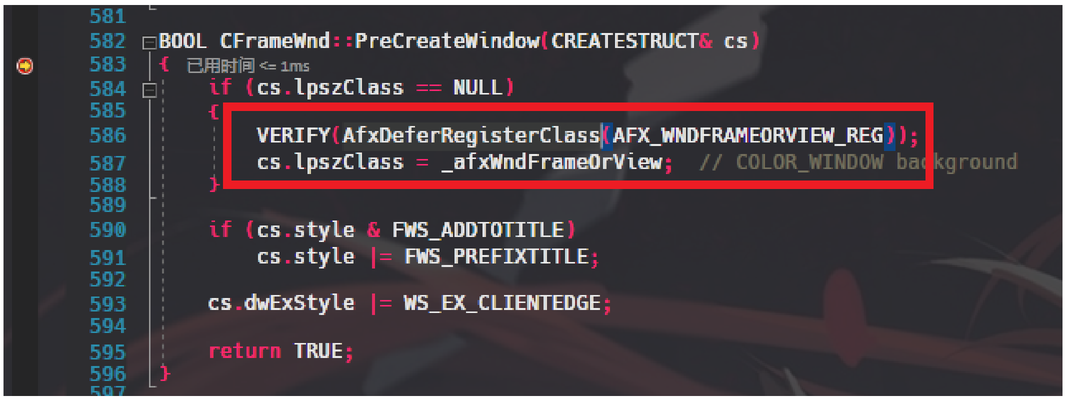


```
CMainFrame::CMainFrame() noexcept
{  已用时间 <= 1ms
    // TODO: 在此添加成员初始化代码
}
```

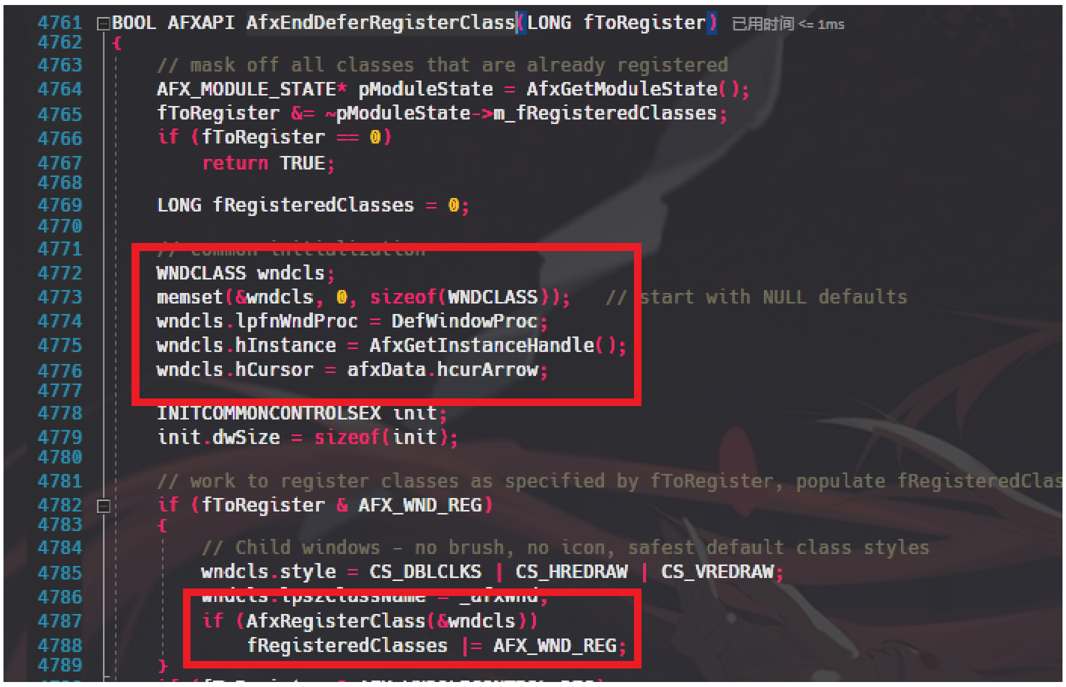


```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{  已用时间 <= 1ms
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: 在此处通过修改
    //  CREATESTRUCT cs 来修改窗口类或样式

    return TRUE;
}
```

5. 在父类`CFrameWnd::PreCreateWindow`中调用了`AfxDeferRegisterClass`函数，其中调用了`AfxDeferRegisterClass`即`AfxEndDeferRegisterClass`

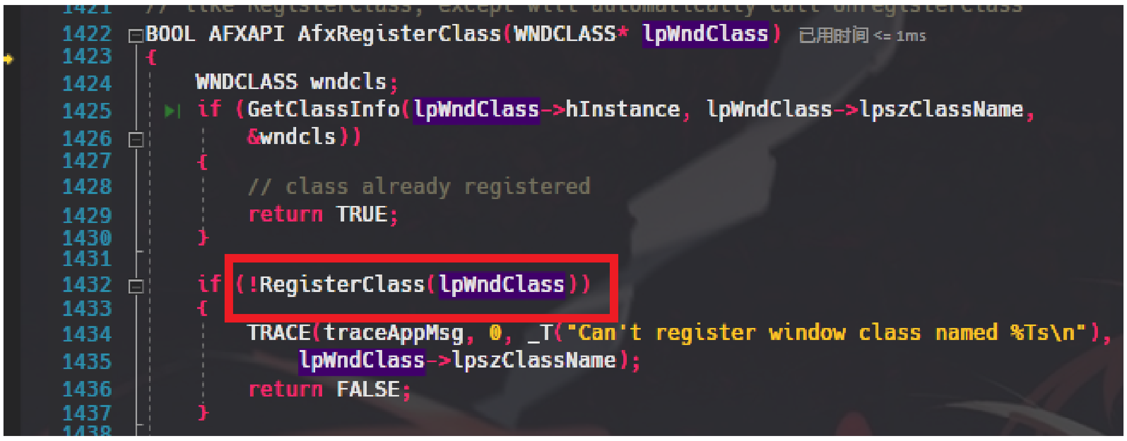


```
BOOL CFrameWnd::PreCreateWindow(CREATESTRUCT& cs)
{  已用时间 <= 1ms
    if (cs.lpszClass == NULL)
    {
        VERIFY(AfxDeferRegisterClass(AFX_WNDFRAMEORVIEW_REG));
        cs.lpszClass = _afxWndFrameOrView;  // COLOR_WINDOW background
    }

    if (cs.style & FWS_ADDTOTITLE)
        cs.style |= FWS_PREFIXTITLE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;

    return TRUE;
}
```



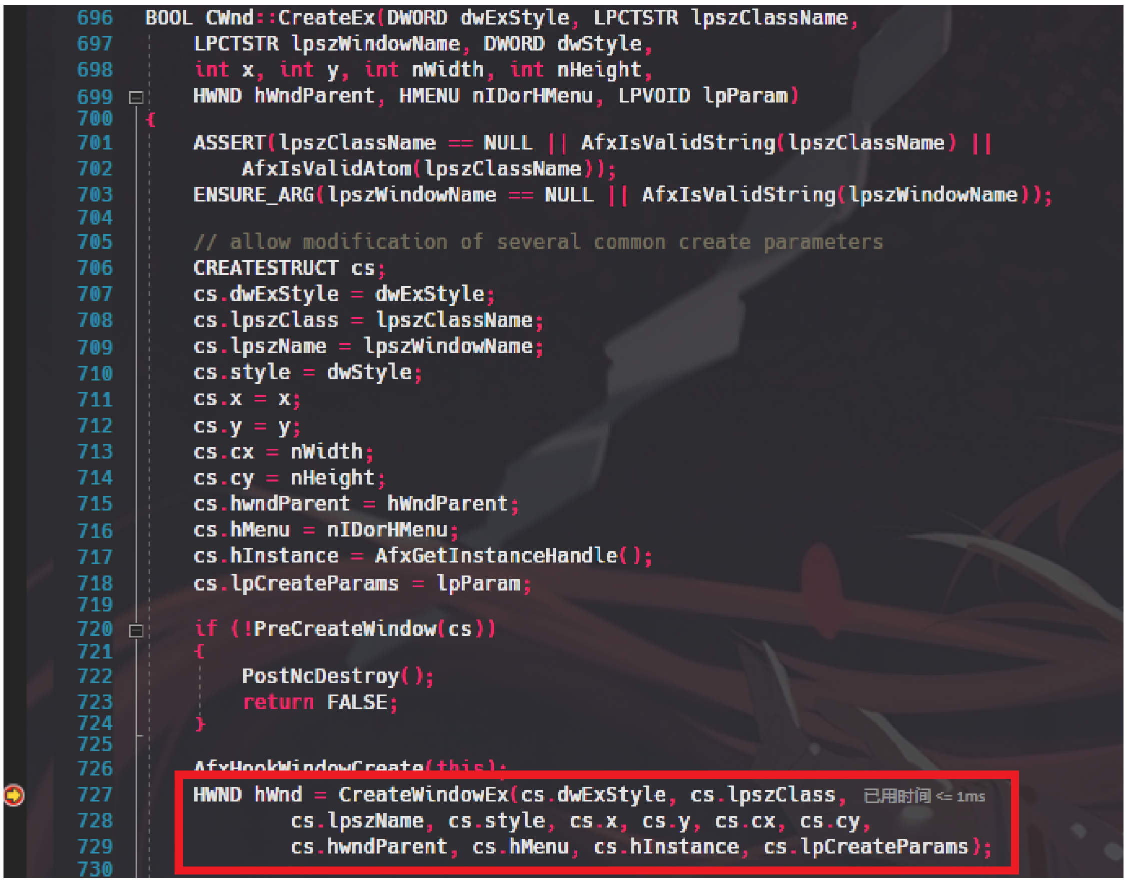
```
BOOL AFXAPI AfxEndDeferRegisterClass(LONG fToRegister)  已用时间 <= 1ms
{
    // mask off all classes that are already registered
    AFX_MODULE_STATE* pModuleState = AfxGetModuleState();
    fToRegister &= ~pModuleState->m_fRegisteredClasses;
    if (fToRegister == 0)
        return TRUE;

    LONG fRegisteredClasses = 0;

    // common initialization
    WNDCLASS wndcls;
    memset(&wndcls, 0, sizeof(WNDCLASS));   // start with NULL defaults
    wndcls.lpfnWndProc = DefWindowProc;
    wndcls.hInstance = AfxGetInstanceHandle();
    wndcls.hCursor = afxData.hcurArrow;

    INITCOMMONCONTROLSEX init;
    init.dwSize = sizeof(init);

    // work to register classes as specified by fToRegister, populate fRegisteredClas
    if (fToRegister & AFX_WND_REG)
    {
        // Child windows - no brush, no icon, safest default class styles
        wndcls.style = CS_DBLCLKS | CS_HREDRAW | CS_VREDRAW;
        wndcls.lpszClassName = _afxWnd;
        if (AfxRegisterClass(&wndcls))
            fRegisteredClasses |= AFX_WND_REG;
    }
```

5. 之后流程转到`AfxRegisterClass`完成窗口类的注册



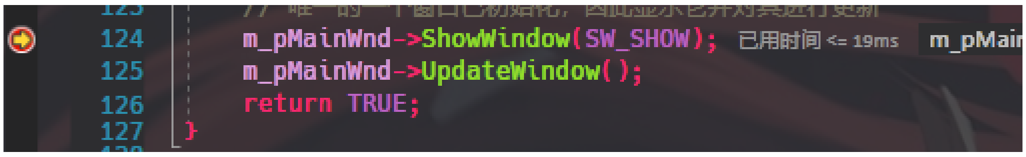
```
// like RegisterClass, except will automatically call UnregisterClass
BOOL AFXAPI AfxRegisterClass(WNDCLASS* lpWndClass)  已用时间 <= 1ms
{
    WNDCLASS wndcls;
    if (GetClassInfo(lpWndClass->hInstance, lpWndClass->lpszClassName,
        &wndcls))
    {
        // class already registered
        return TRUE;
    }

    if (!RegisterClass(lpWndClass))
    {
        TRACE(traceAppMsg, 0, _T("Can't register window class named %Ts\n"),
            lpWndClass->lpszClassName);
        return FALSE;
    }
```

6. 流程来到`CWnd::CreateEx`通过`CreateWindowsEx`创建窗口



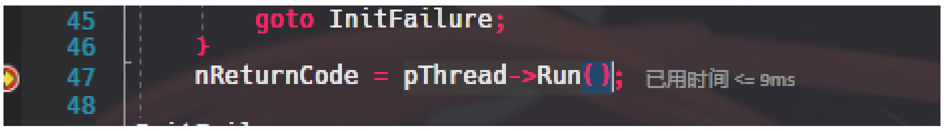
```
BOOL CWnd::CreateEx(DWORD dwExStyle, LPCTSTR lpszClassName,
    LPCTSTR lpszWindowName, DWORD dwStyle,
    int x, int y, int nWidth, int nHeight,
    HWND hWndParent, HMENU nIDorHMenu, LPVOID lpParam)
{
    ASSERT(lpszClassName == NULL || AfxIsValidString(lpszClassName) ||
        AfxIsValidAtom(lpszClassName));
    ENSURE_ARG(lpszWindowName == NULL || AfxIsValidString(lpszWindowName));

    // allow modification of several common create parameters
    CREATESTRUCT cs;
    cs.dwExStyle = dwExStyle;
    cs.lpszClass = lpszClassName;
    cs.lpszName = lpszWindowName;
    cs.style = dwStyle;
    cs.x = x;
    cs.y = y;
    cs.cx = nWidth;
    cs.cy = nHeight;
    cs.hwndParent = hWndParent;
    cs.hMenu = nIDorHMenu;
    cs.hInstance = AfxGetInstanceHandle();
    cs.lpCreateParams = lpParam;

    if (!PreCreateWindow(cs))
    {
        PostNcDestroy();
        return FALSE;
    }

    AfxHookWindowCreate(this);
    HWND hWnd = CreateWindowEx(cs.dwExStyle, cs.lpszClass,  已用时间 <= 1ms
            cs.lpszName, cs.style, cs.x, cs.y, cs.cx, cs.cy,
            cs.hwndParent, cs.hMenu, cs.hInstance, cs.lpCreateParams);
```

7. 之后流程回到`theApp->InitInstance()`，调用`ShowWindow`和`updateWindow`显示更新窗口



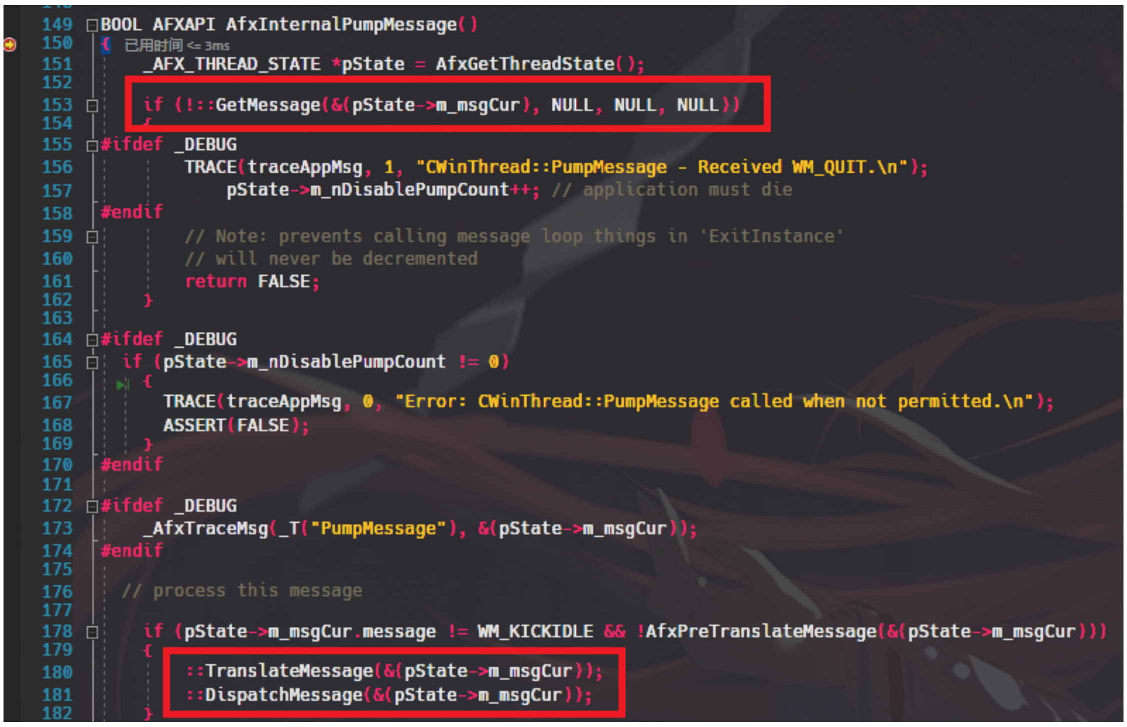
```
    // 唯一的一个窗口已初始化，因此显示它并对其进行更新
    m_pMainWnd->ShowWindow(SW_SHOW);  已用时间 <= 19ms   m_pMair
    m_pMainWnd->UpdateWindow();
    return TRUE;
}
```

8. 之后回到`AfxWinMain`函数，调用`pThread->Run()`进入消息循环



```
            goto InitFailure;
        }
        nReturnCode = pThread->Run();  已用时间 <= 9ms
```

9. 在`run`方法中调用了`AfxInternalPumpMessage`，消息循环如下



```
BOOL AFXAPI AfxInternalPumpMessage()
{  已用时间 <= 3ms
    _AFX_THREAD_STATE *pState = AfxGetThreadState();

    if (!::GetMessage(&(pState->m_msgCur), NULL, NULL, NULL))
    {
#ifdef _DEBUG
        TRACE(traceAppMsg, 1, "CWinThread::PumpMessage - Received WM_QUIT.\n");
            pState->m_nDisablePumpCount++; // application must die
#endif
        // Note: prevents calling message loop things in 'ExitInstance'
        // will never be decremented
        return FALSE;
    }

#ifdef _DEBUG
  if (pState->m_nDisablePumpCount != 0)
    {
      TRACE(traceAppMsg, 0, "Error: CWinThread::PumpMessage called when not permitted.\n");
      ASSERT(FALSE);
    }
#endif

#ifdef _DEBUG
    _AfxTraceMsg(_T("PumpMessage"), &(pState->m_msgCur));
#endif

  // process this message

    if (pState->m_msgCur.message != WM_KICKIDLE && !AfxPreTranslateMessage(&(pState->m_msgCur)))
    {
        ::TranslateMessage(&(pState->m_msgCur));
        ::DispatchMessage(&(pState->m_msgCur));
    }
```