Windows API开发Windows程序

Windows API ——

Windows Application Programming Interface

Windows 应用程序接口，是微软提供的、开发Windows程序所用的函数，以SDK——Software Development Kit（软件开发包）的形式发行。

安装visual studio时，已经安装了SDK。

windows程序运行与开发的特点

windows程序（视窗程序）与大一学习的控制台程序在运行机制和开发模式上有着很大的差别。运行机制的特点：

- 每一个程序都有“窗口”，（无论可见或者不可见），多数时候“窗口”还不止一个（主“窗口”、子“窗口”），用户同程序交互操作，就是通过鼠标、键盘操作各种“窗口”来进行的。譬如，鼠标点击最小化按钮，使程序界面最小化。
- 程序里涉及的所有窗口都交给windows系统去创建、消亡，windows会给我们窗口的标识——句柄，通过它我们可以通知windows对“窗口”做什么工作。譬如，程序退出了，我们就通知windows让“窗口”消亡，同时windows也会通知程序：就要退出了，还有什么工作要做，没有的话就撤销所有资源了...等等。可见，有了windows操作系统“从中作梗”，一个程序运行起来一定是“消息满天飞”。
- Windows程序间传递的通知称为“消息”——message；对消息做出回应或响应的代码，通常写在函数里，称之为“消息响应函数”；windows系统中预定义（define）了若干消息，我们也可以自定义消息。“窗口”不是唯一能够接受和处理消息的windows资源，譬如，MFC封装的线程类，它可以接受线程消息。

windows程序运行与开发的特点

- 消息

消息有唯一的标识符，需要预先定义好。windows系统内部就预先定义了一系列消息，形如

```
#define    WM_PAINT        0x000F
```

消息名　　消息的值

可见，消息的标识其实就是一个整数。消息类型与值的对应关系，如图所示。

值范围	消息意义
0x0001~0x0087	主要是窗口消息
0x00A0~0x00A9	非客户区消息
0x0100~0x0108	键盘消息
0x0111~0x0126	菜单消息
0x0132~0x0138	颜色控制消息
0x0200~0x020A	鼠标消息
0x0211~0x0213	菜单循环消息
0x0220~0x0230	多文档消息
0x03E0~0x03E8	DDE消息
0x0400	WM_USER
0x0400~0x7FFF	自定义消息

windows程序运行与开发的特点

- 消息的数据结构

在windows中，消息的传递涉及的不仅仅是它的标识，还有窗口、参数、位置等等，所以，windows将消息相关的数据用一个结构体封装起来。

消息的发送和接受都是通过windows提供的函数进行操作的，图中的结构常作为函数的参数，或者函数的参数是结构体成员中的几个，函数内部再对其进行处理成结构体变量。

图中数据结构是windows预先定义好的（WinUser.h），包含Windows.h就可以了。我们在编程时会经常用到系统提供的数据结构和函数，它们都在一种叫SDK的开发包中。

visual studio在安装时一并安装了SDK。

```
typedef  struct  tagMSG
{
    HWND      hwnd;        //窗口句柄
    UINT      message;     //消息标识
    WPARAM    wParam;      //参数1
    LPARAM    lParam;      //参数2
    DWORD     time;        //时间值
    POINT     pt;          //位置（鼠标）
}MSG;
```

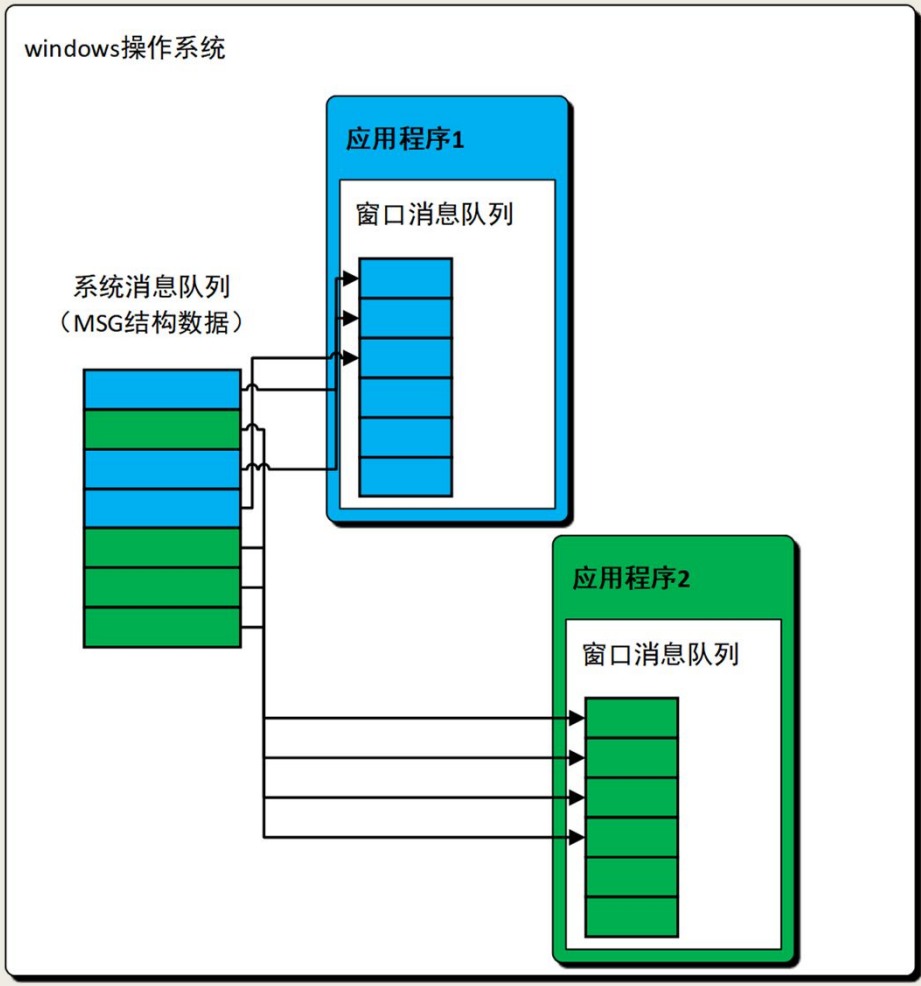
windows程序运行与开发的特点

- 消息运作机制

消息通常需要通过windows的消息队列中转：先到操作系统的消息队列，然后根据窗口句柄发给相应窗口。

消息可以发给自己、也可以发给其他程序，都需要操作系统参数中转（窗口句柄可以识别消息到底发给哪个窗口）。

程序内部有专门的函数去读取消息、处理消息。



基于windows API的程序开发（简介）

先看看Windows基本数据类型，基本都是利用typedef改名后的基本数据类型。

WORD:	16位无符号整形数据
DWORD:	32位无符号整型数据
DWORD32:	32位无符号整型数据
DWORD64:	64位无符号整型数据
INT:	32位有符号整型数据类型
INT32(__int32):	32位有符号整型
INT64(__int64):	64位有符号整型
INT_PTR:	INT32或INT64
UINT:	无符号INT
LONG:	32位符号整型（LONG32）
ULONG:	无符号LONG
LONGLONG:	64位符号整型（LONG64）
SHORT:	无符号短整型（16位）
LPARAM:	消息的L参数（ LONG_PTR ）
WPARAM:	消息的W参数（ UINT_PTR ）
HANDLE:	对象的句柄，最基本的句柄类型
HICON:	图标的句柄
HINSTANCE:	程序实例的句柄

HKEY:	注册表键的句柄
HMODULE:	模块的句柄
HWND:	窗口的句柄
LPSTR:	字符指针，也就是字符串变量
LPCSTR:	字符串常量
LPCTSTR:	LPCWSTR或LPCSTR类型（根据环境）
LPCWSTR:	UNICODE字符串常量
LPDWORD:	指向DWORD类型数据的指针
CHAR:	8比特字节
TCHAR:	WCHAR或为CHAR（根据环境）
UCHAR:	无符号CHAR
WCHAR:	16位Unicode字符
BOOL:	布尔型变量
BYTE:	字节类型（8位）
CONST:	常量
FLOAT:	浮点数据类型
SIZE_T:	ULONG或ULONGLONG（根据环境）
VOID:	无类型，相当于标准C语言中的void

基于windows API的程序开发（简介）

Windows API中类型名称的基本规律：

- 基本数据类型包括：BYTE、CHAR、WORD、SHORT、INT等。
- 指针类型的命令方式一般是在其指向的数据类型前加“LP”或“P”，比如指向DWORD的指针类型为“LPDWORD”和“PDWORD”。 LP”或“P”的存在的缘由是win32程序中兼容win16程序的需要。这是历史遗留问题。
- 各种句柄类型的命名方式一般都是在对象名前加“H”，比如位图（BITMAP）对应的句柄类型为“HBITMAP”。
- 无符号类型一般是以“U”开头，比如“INT”是符号类型，“UINT”是无符号类型。

基于windows API的程序开发（简介）

函数的调用约定

在调用API函数、编写DLL导出函数时，必须对函数的参数入栈顺序和清理进行相关约定。所以，在此类函数的定义或原型中会进行调用约定处理。

如：APIENTRY、WINAPI、CALLBACK、__stdcall等。其实，它们是同一个约定：

- 参数从右向左压入堆栈
- 函数被调用者修改堆栈
- 函数名(在编译器这个层次)自动加前导的下划线，后面紧跟一个@符号，其后紧跟着参数的尺寸

这些调用约定，仅在需要时加入（如：DLL导出函数、接口函数、回调函数等），其它不用考虑。

基于windows API的程序开发（简介）

```
int APIENTRY wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
{
/*********************************************************************/
/*  1、注册。MyRegisterClass函数
    目的:   设定窗口的样式（菜单、工具栏等）;
            把消息循环的处理函数的指针传递给操作系统*/
/*********************************************************************/

    MyRegisterClass(hInstance);

/*********************************************************************/
/*  2、InitInstance函数。调用API函数 CreateWindow,创建窗口
    并根据nCmdShow的值控制窗口是显示 ，最大化显示、最小化显示，还是隐藏等*/
/*********************************************************************/
    if (!InitInstance (hInstance, nCmdShow))
        return FALSE;

    //加载Resource view中对应的快捷键资源，这样，程序就可以用快捷键了。
    //前提是，你都得设置。当然，也可以不用快捷键。
    HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_EX01));

/*********************************************************************/
/*  3、主程序进入消息循环
当程序退出时，会Get到WM_QUIT消息，此时GetMessage就会返回0。所以，它不是死循环！！！！
只要不退出，对本程序窗口的所有操作，都会被操作系统获得，并添加到本程序的消息队列中。
GetMessage就是不停的从消息队列获取消息、传导到WndProc完成响应、再GetMessage....*/
/*********************************************************************/
    MSG msg;    //MSG结构体变量，存储消息用的
    while (GetMessage(&msg, NULL, 0, 0))
    {   //处理快捷键消息：它通过Resource view中设定的"键-函数"对应表直接完成。
        if( TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
            continue;
        //如果不是，则传导消息到WndProc函数去处理
        TranslateMessage(&msg);     //处理键盘消息
        DispatchMessage(&msg);      //经由此函数，调用WndProc，完成消息的响应。
    }

    //4、主程序退出，程序结束.
    return (int) msg.wParam;
}
```

基于windows API的程序开发（简介）

```
58  ATOM MyRegisterClass(HINSTANCE hInstance)
59  {
60      //WNDCLASSEX是windows定义的传递参数的结构体
61      WNDCLASSEX wcex;
62      wcex.cbSize = sizeof(WNDCLASSEX);
63      wcex.style          = CS_HREDRAW | CS_VREDRAW;
64
65      //把消息处理函数的指针(就是函数名)传递给操作系统
66      wcex.lpfnWndProc    = WndProc;
67      wcex.cbClsExtra     = 0;
68      wcex.cbWndExtra     = 0;
69
70      //这个是操作系统产生、分配给本程序的，并通过参数传递进来的那个
71      wcex.hInstance      = hInstance;
72
73      //程序的图标，显示在左上角的那个
74      wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_EX01));
75
76      //设定鼠标样式，如果鼠标移动到此窗口时，将会按照此设定显示
77      wcex.hCursor        = LoadCursor(NULL, IDC_ARROW);
78
79      //窗口客户区的背景颜色，这里用来预定义的颜色
80      wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
81
82      //设定菜单，通过Resource view可以调整
83      wcex.lpszMenuName   = MAKEINTRESOURCE(IDC_EX01);
84
85      //给起个名字。
86      //后面在API函数 CreateWindow时要根据这个名字找这个注册的信息的，两个要一样的。
87      wcex.lpszClassName  = L"起个好名字";
88
89      //程序的图标，程序最小化到工具栏是显示的那个
90      wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
91
92      //RegisterClassEx是windows API函数
93      //正如其名称，这些参数将保存到windows进程管理中
94      return RegisterClassEx(&wcex);
95  }
```

基于windows API的程序开发（简介）

创建并显示窗口。也是调用windows API函数来完成的。

```
109 BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
110 {
111     HWND hWnd;   //窗口句柄
112
113     hInst = hInstance;
114
115     hWnd = CreateWindow(
116         L"起个好名字",          //WNDCLASSEX名字
117         L"Ex01",                //显示窗口标题栏中字符串
118         WS_OVERLAPPEDWINDOW, //显示风格
119         100,                    //左上角位置的横坐标（像素）
120         50,                     //左上角位置的纵坐标
121         400,                    //窗口宽度（像素）
122         300,                    //窗口高度
123         NULL,                   //父窗口句柄
124         NULL,                   //菜单句柄。WNDCLASSEX已经设好了
125         hInstance,              //操作系统分配的句柄
126         NULL);                  //传递给程序的其它参数
127
128     if (!hWnd)
129         return FALSE;
130
131     //显示窗口
132     ShowWindow(hWnd, nCmdShow);
133     UpdateWindow(hWnd);
134
135     return TRUE;
136 }
```

基于windows API的程序开发（简介）

消息处理函数，也叫消息响应函数。

```
148 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
149 {
150     int wmId, wmEvent;
151     PAINTSTRUCT ps;
152     HDC hdc;
153     WCHAR text[] = L"输出文字啦！";
154     RECT txRect = {40, 100, 360, 150};
155
156     switch (message)
157     {
158     case WM_COMMAND:        //菜单命令
159         wmId    = LOWORD(wParam);
160         wmEvent = HIWORD(wParam);
161         // Parse the menu selections:
162         switch (wmId)
163         {
164         case IDM_ABOUT:
165             DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
166             break;
167         case IDM_EXIT:
168             DestroyWindow(hWnd);
169             break;
170         default:
171             return DefWindowProc(hWnd, message, wParam, lParam);
172         }
173         break;
174     case WM_PAINT:      //绘制客户区内容
175         hdc = BeginPaint(hWnd, &ps);
176         DrawTextW(hdc, text, wcslen(text), &txRect, DT_CENTER);
177         EndPaint(hWnd, &ps);
178         break;
179     case WM_DESTROY:
180         PostQuitMessage(0);
181         break;
182     default:
183         return DefWindowProc(hWnd, message, wParam, lParam);
184     }
185     return 0;
186 }
```

基于windows API的程序开发（简介）

总结：

1、windows程序的开发，是一门技术；

2、它的开发过程是与windows操作系统密切结合在一起的；

3、“消息”是windows程序运转的基石。

有关windows API的参考资料：

《windows核心编程》

《windows系统编程》

基于windows API的程序开发（简介）

关于stdafx.h和stdafx.cpp

在visual C++向导生成的程序中，这两个文件是“标配”。多数时候，没有它们编译会报错。它们的作用是什么呢?

C++开发的程序常常为多文件形式的，几十或上百个文件在中小规模的windows程序中是很常见的。为提高编译速度，visual C++开发环境中为那些不经常更改、相对固定的文件进行标识。既然文件中代码没有变，那就不需要重新编译。这些代码的头文件通常被放到stdafx.h中，而且，visual C++编译器规定，只要在CPP文件中出现stdafx.h之前的代码，统统不需要重新编译，仅仅链接其已经编译过的代码（保存在.pch中）。这也是为什么我们看到的CPP文件中的开始部分总有一句#include “stdafx.h”的原因。

所以:

1、windows、库文件（如MFC）或者第三方库文件，其头文件可以放在stdafx.h中；2、我们自己编写的、相对固定的代码，也可以把其头文件放在stdafx.h中；

3、cpp文件中，需要加#include语句的，请加在#include “stdafx.h”这句的下面。