窗口的创建和销毁

SDK中的错误处理

`GetLastError`API可以获取最近一次的错误码，在监视窗口输入`@err,hr`或者`*(int *)(tib + 0x34),hr`也可以`FormatMessage`可以格式化错误码

窗口风格

- 三种基本风格
  - WS_OVERLAPPED，重叠窗口
  - WS_POPUP，弹出窗口
  - WS_CHILD，子窗口

窗口创建与销毁

1. 设计注册窗口类
   - `RegisterClass`用于注册一个窗口类

```
{
    // ...
    WNDCLASS wc;    // 窗口类结构

    // 初始化窗口类结构.
    wc.style = CS_HREDRAW | CS_VREDRAW; // 窗口风格
    wc.lpfnWndProc = (WNDPROC) MainWndProc; // 窗口过程函数
    wc.cbClsExtra = 0; // 在窗口类结构中额外扩展空间
    wc.cbWndExtra = 0; // 在窗口实例中额外扩展空间
    wc.hInstance = hinstance; // 应用程序实例句柄
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION); // 窗口图标
    wc.hCursor = LoadCursor(NULL, IDC_ARROW); // 窗口光标
    wc.hbrBackground = GetStockObject(WHITE_BRUSH); // 窗口背景
    wc.lpszMenuName =  "MainMenu"; // 窗口菜单名
    wc.lpszClassName = "MainWindowClass"; // 窗口类名

    // 注册窗口
    if (!RegisterClass(&wc))
       return FALSE;
    // ...
}
```

2. 创建窗口实例

- `CreateWindow`用于创建窗口

```
{
    // 注册窗口类...
    // ...

    // 创建窗口
    HWND  hwnd = CreateWindow(
    "MainWClass",         // 窗口类名
    "Sample",             // 标题栏名
    WS_OVERLAPPEDWINDOW, // 创建窗口风格
    CW_USEDEFAULT,        // x坐标
    CW_USEDEFAULT,        // y坐标
    CW_USEDEFAULT,        // 宽
    CW_USEDEFAULT,        // 高
    (HWND) NULL,          // 父窗口
    (HMENU) NULL,         // 菜单
    hinstance,            // 应用实例句柄
    (LPVOID) NULL);       // 窗口创建数据，CREATESTRUCT结构体

    if (!hwnd)
    return FALSE;
    // ...
}
```

3. 显示窗口

- `ShowWindow`用于显示窗口

```
{
    // 注册窗口类...
    // 创建窗口...
    // ...

    // 显示窗口
    ShowWindow(hwnd, nCmdShow);
    // ...
}
```

4. 更新窗口

- `UpdateWindow`用于更新窗口

```
{
    // 注册窗口类...
    // 创建窗口...
    // 显示窗口
    // ...

    // 更新窗口
    UpdateWindow(hwnd);
    // ...
}
```

5. 实现消息循环
    - `MSG`结构

```
typedef struct tagMSG {
HWND        hwnd;     // 窗口句柄
UINT        message;  // 消息编号
WPARAM      wParam;   // 参数1，在消息中有用
LPARAM      lParam;   // 参数2，在消息中有用
DWORD       time;     // 时间
POINT       pt;       // 鼠标位置
} MSG, *PMSG, NEAR *NPMSG, FAR *LPMSG;
```

    - `GetMessage`从消息队列中取得消息

```
    {
        // 注册窗口类...
        // 创建窗口...
        // 显示窗口...
        // 更新窗口...
        // ...

        // 消息循环
        MSG msg;
        while(GetMessage(&msg, NULL, 0, 0)) {
            TranslateMessage(&msg);  // 翻译消息
            DispatchMessage(&msg);       // 投递消息
        }
    }
```

6. 实现窗口过程

```
LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg) {
        case WM_CLOSE:
            DestroyWindow(hwnd);
```

```
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);      // 投递退出消息
            return 0;
         //...
        default:
            break;
    }
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}
```