2020/06/19_SDK_第1课_SDK入门(基础概念)

笔记本: SDK编程
创建时间: 2020/6/19 星期五 15:46
作者: ileemi

第二阶段课程介绍

- SDK（5个课时）
- MFC（12课时）
- windows（12课时）
- 网络（6课时）
- 数据库（6课时）
- com（5课时）

API：Application Programming Interface -- 应用程序编程接口
SDK：Software Development Kit -- 软件开发工具包
MFC：Microsoft Foundation Classes -- 微软基础类

SDK课程介绍

SDK -- software development kit 软件开发包
一组功能性函数 + 说明文档 + 介绍 + demo

CreatFile

- 基础概念
- **消息机制** -- 注意：需要了解Windows消息机制背后的**运行机制**
- 常用消息
- 资源和对话框
- 常用控件的使用

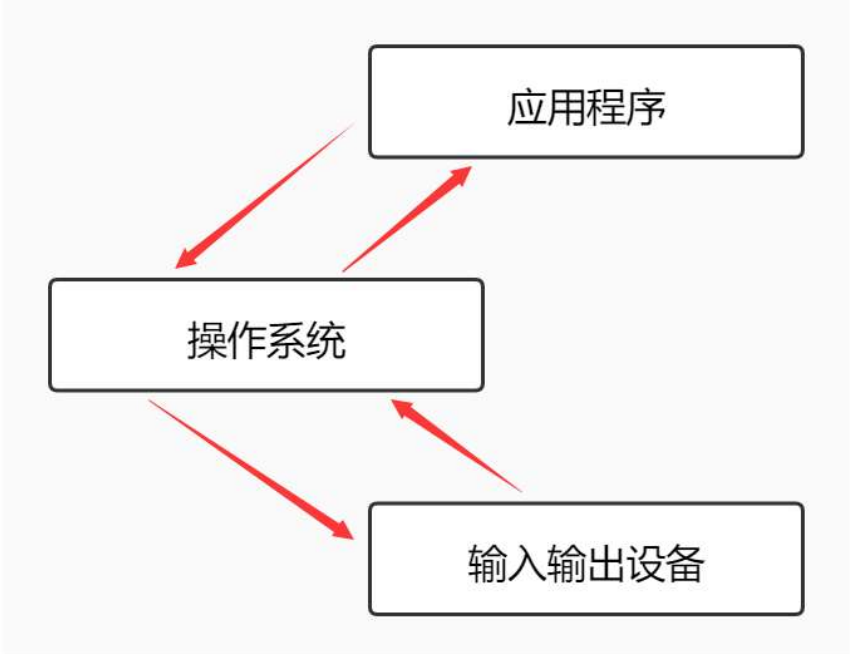
SDK 基础概念

API

API -- application programing interface 应用程序编程接口

控制台是DOS的输出设备

输入输出设备：
输入：键盘、鼠标、打印机等
输出：显示器等



如上图所示，输入输出设备被操作系统进行了接管，应用程序不能直接控制输入输出设备，**操作系统允许应用程序使用操作系统提供的API（操作系统提供的一组功能函数）来控制输入输出设备。**

内核对象

内核对象：受保护的对象

当不希望直接去操作操作系统中的一些对象时，需要通过提供的API去进行操，这样的对象被称为受保护对象（内核对象）。

操作系统的内核对象有：鼠标、键盘等都有属于自己的内核对象，需要使用对应的API进行操作。

用户和内核模式

0环 3环
内核 用户

- 80386芯片的4个权限级别（最高到最低）：0~3（0、1、2、3）
- Windows系统所使用的两个权限级别（Windows只使用了0环和3环--最高和最低）：0、3
  操作系统运行在0环（内核层），用户发开的程序运行在3环（用户层），用户层无法访问内核层（操作系统层）反之，操作系统可以访问用户层。
- 用户模式下的限制

低权限不能访问高权限

DOS 系统没有分层，所以当时年代很容易被攻击

句柄

操作系统中有很多的内核对象，如果想修改或者访问、操作等，怎样能确定访问的就是自己想要修改的，这个时候需要通过对应的 "ID"，类似于学生管理系统中对指定ID的学生进行信息操作一样。

句柄：学生管理中的 "ID"

消息机制

鼠标点击记事本菜单，之后对其操作出现的想用是由记事本（软件）决定的。
鼠标点击是时操作系统要比应用程序实现直到，但是两者是如何快速的对其做出相应处理的?

通过函数指针（回调函数）

具体实现例如：
当鼠标操作过来的时候，应用程序将对应的处理函数的地址告诉操作系统，让鼠标消息来的时候，操作系统直接去通过对应的地址调用这个函数，以做到及时响应。

对于不同的输入操作系统会对其进行封装，封装成一个结构体，表示输入的种类，这个结构体称为**消息结构体**

操作系统通过调用 回调函数 将消息传入到消息结构体内，之后让应用程序对其进行相应的处理，这一套东西称为**消息机制**。

Windows是消息驱动的操作系统。

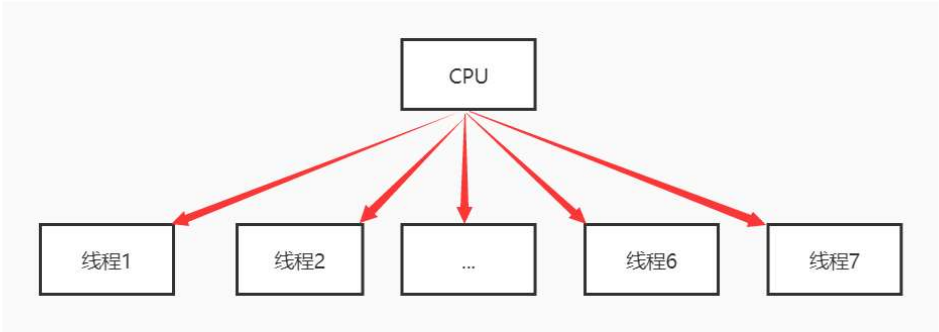
一般的窗口程序我们称之为消息（各种各样的输入）驱动。

WINDOWS 多任务的实现

控制台程序 -- 单任务（Dos系统同样也是单任务），调试程序是代码一行一行进行调试

Windows 可以多行同时进行

多任务操作体统，进程间可以来回切换

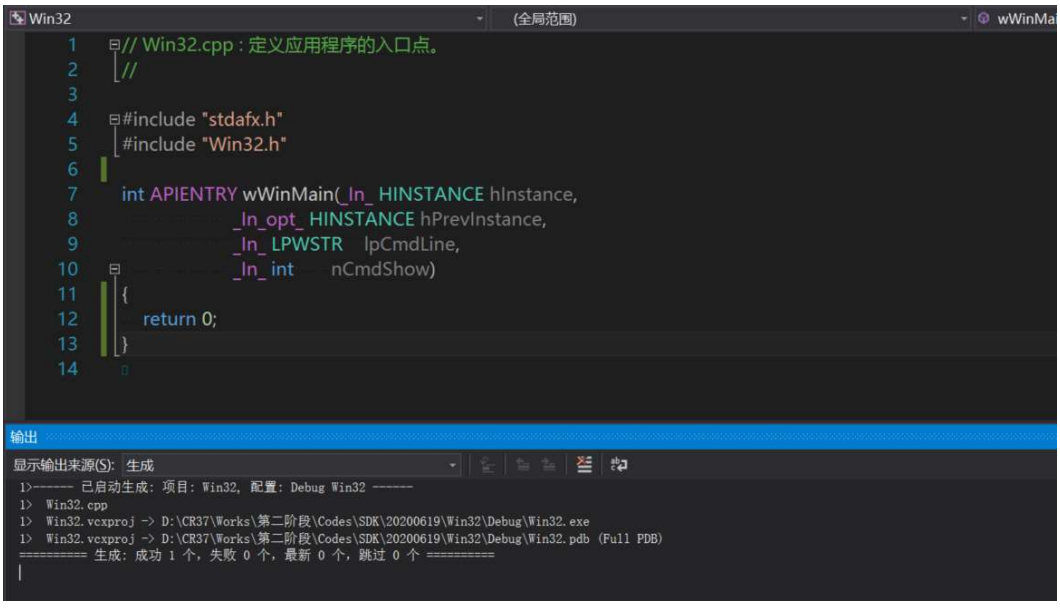


Windows VS Console

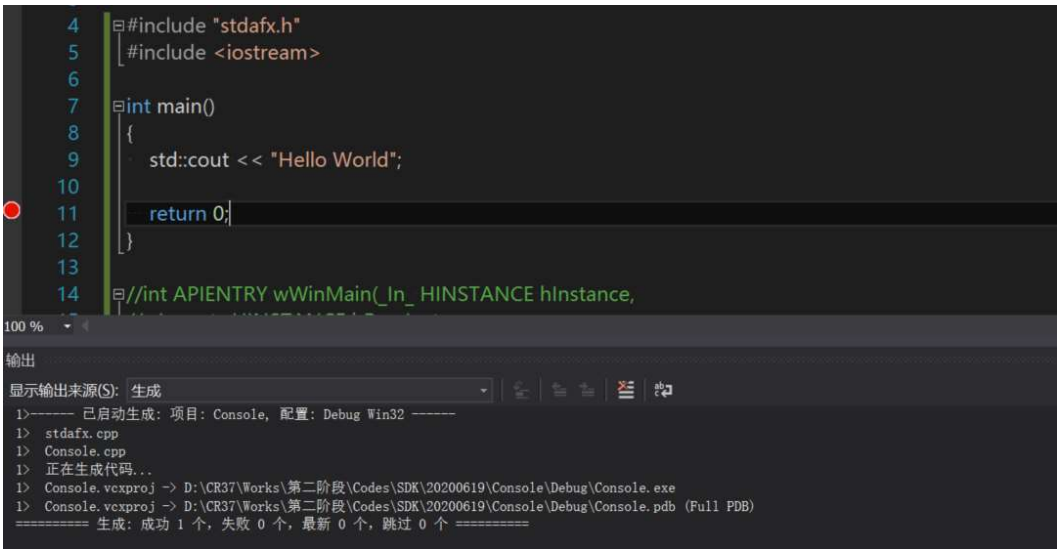
- 入口函数
- 链接方式

入口函数

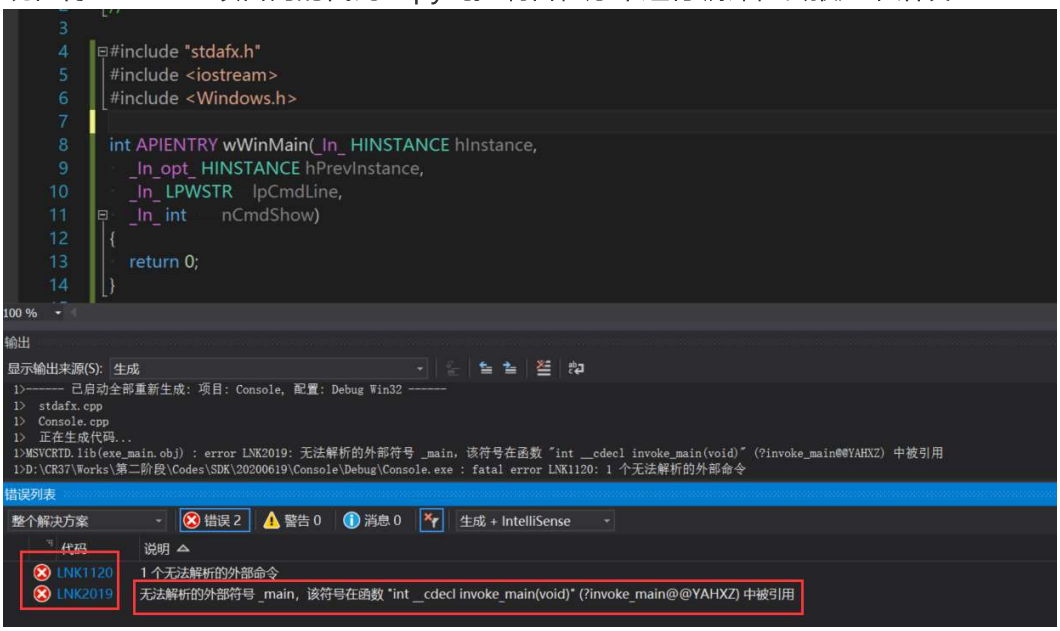
Windows 入口函数代码示例：



Consoal控制台程序：

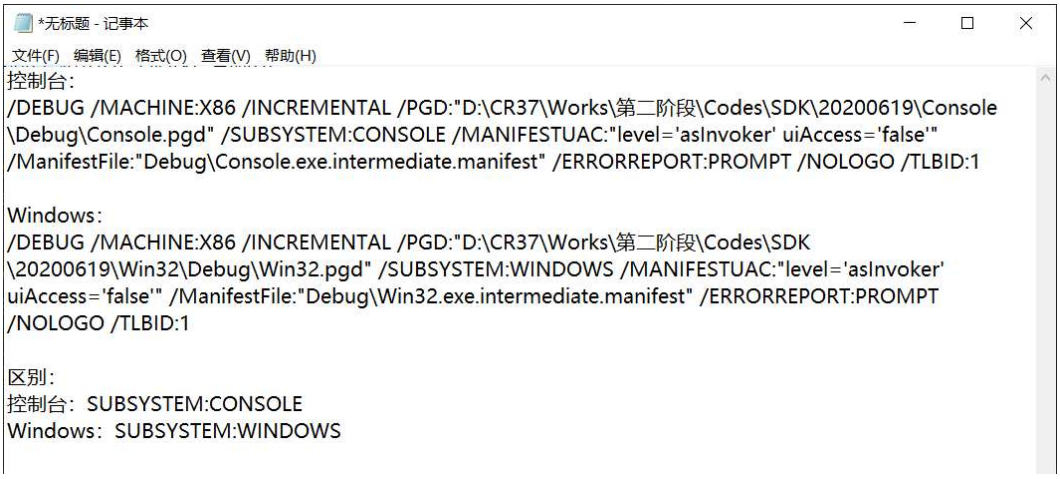


现在将Windows项目内的代码copy到控制台程序中进行编译，会报如下错误：

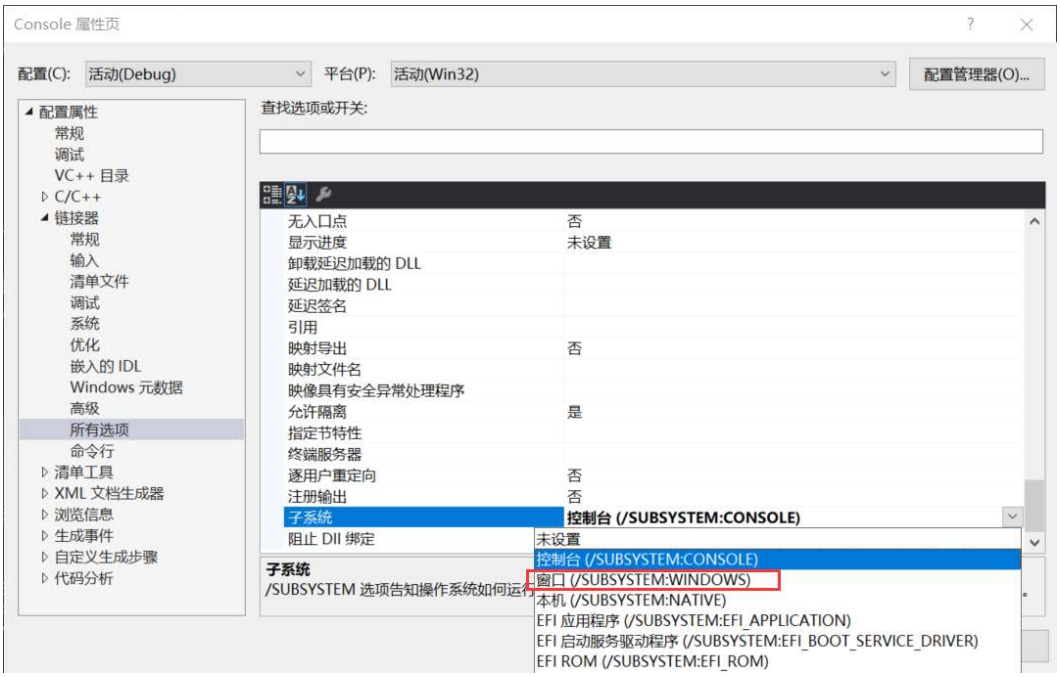


通过错误提示，可以看出是一个链接错误，于是，分别将两个程序的链接选项拷贝出来进行对比：

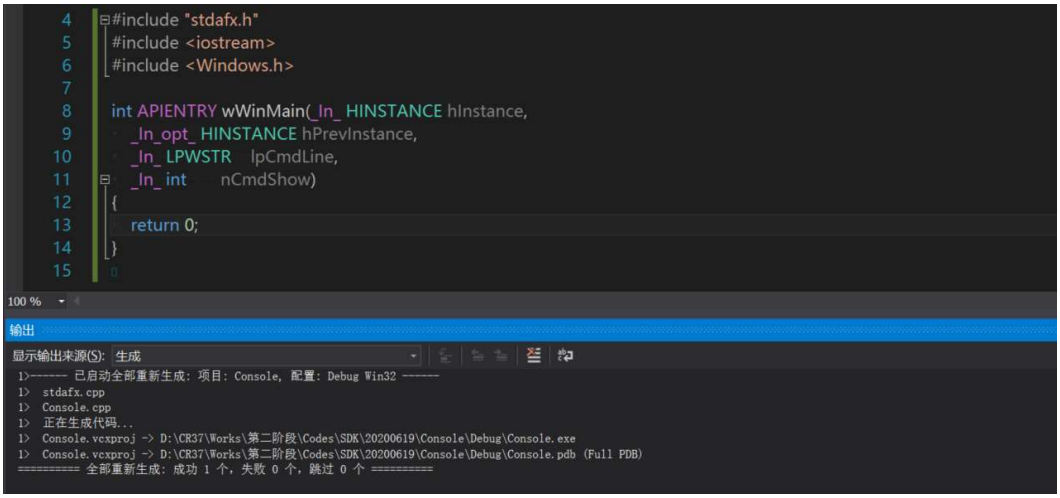
两者区别如下图所示：



尝试将控制台的选项 SUBSYSTEM:CONSOLE 修改为 SUBSYSTEM:WINDOWS



修改后再次编译效果图如下:



结论:

Windows和控制台入口点区别:

控制台: SUBSYSTEM:CONSOLE

Windows: SUBSYSTEM:WINDOWS

Windows入口函数以及参数

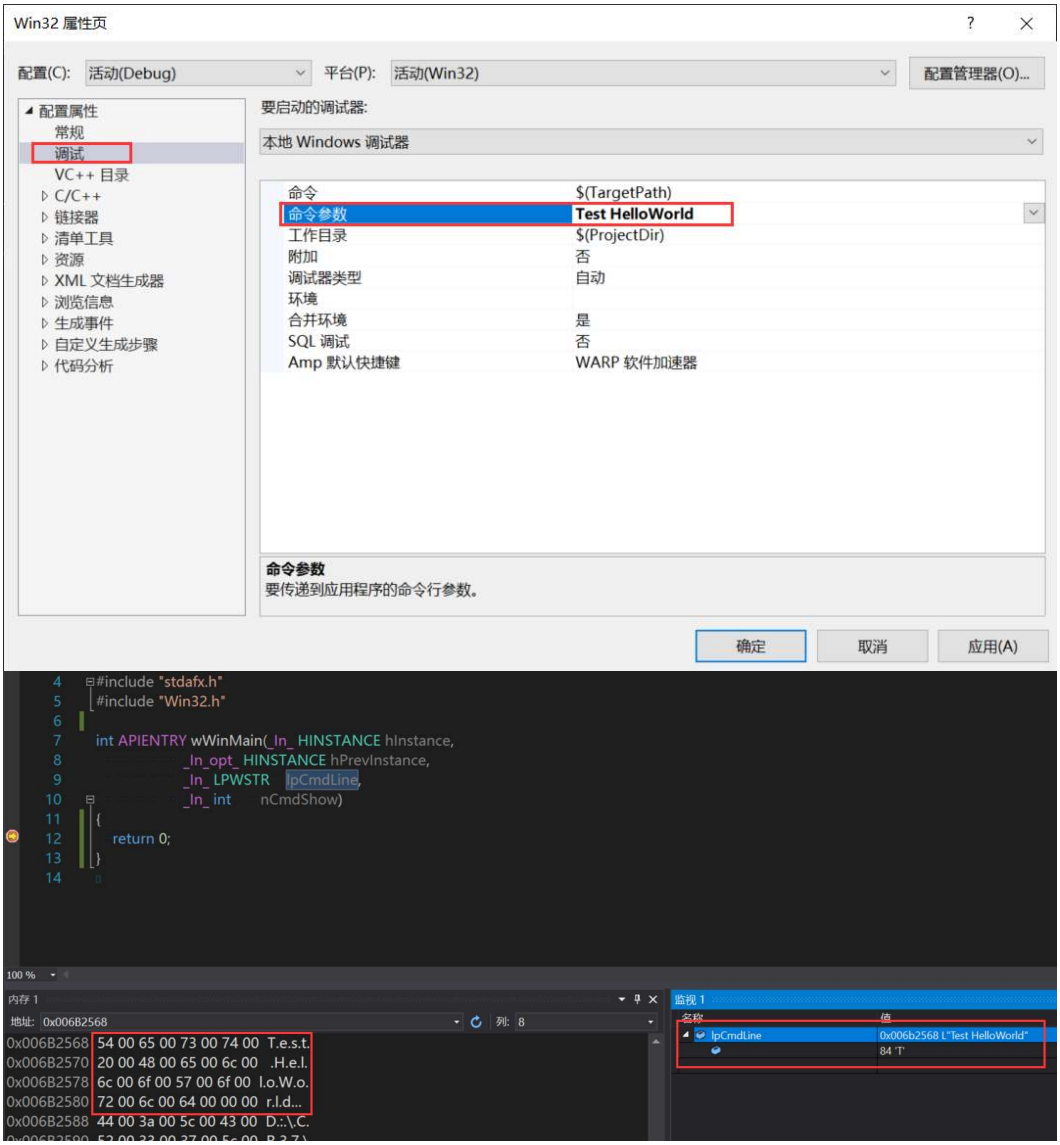
```
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
    _In_opt_ HINSTANCE hPrevInstance,
    _In_ LPWSTR lpCmdLine,
    _In_ int nCmdShow)
{
    return 0;
}
```

_In_: 空宏 说明性宏，注释宏

HINSTANCE hInstance: 实例句柄，代表应用程序本身

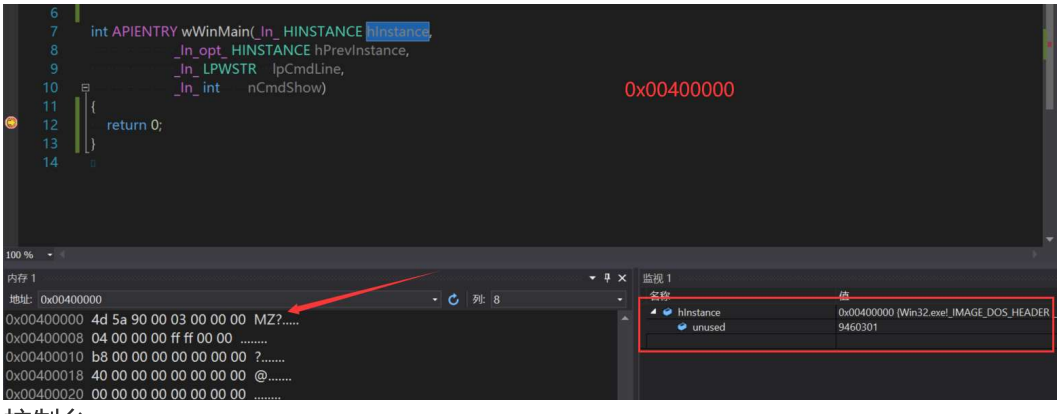
```
HINSTANCE hPrevInstance：为了兼容老的入口函数，现在程序跑起来后为空
LPWSTR lpCmdLine：命令行参数(项目-属性-属性配置-调试-命令行参数)
int nCmdShow：控制显示
```

VS 为程序添加命令行参数：

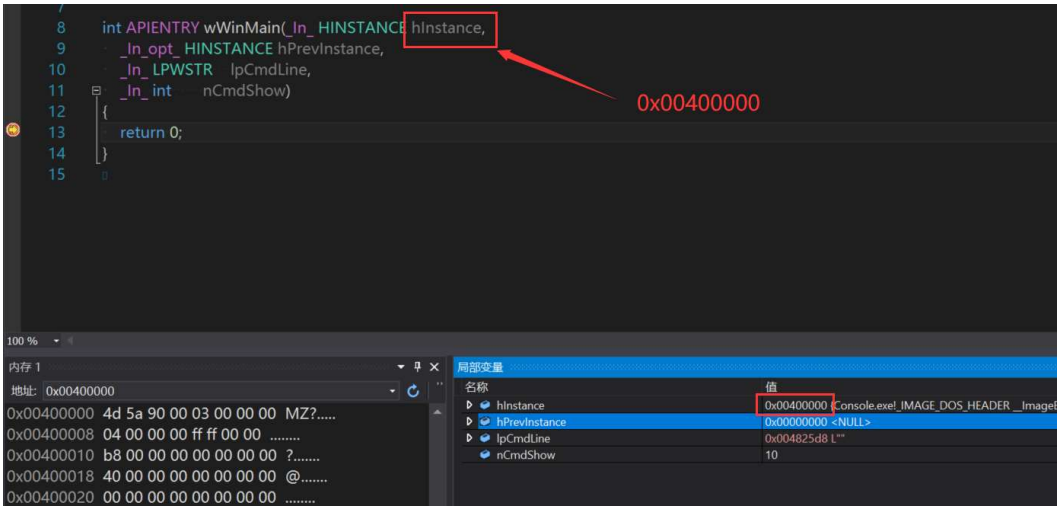


hInstance 句柄特殊之处

Windows：0x00400000 进程中的首地址



控制台：

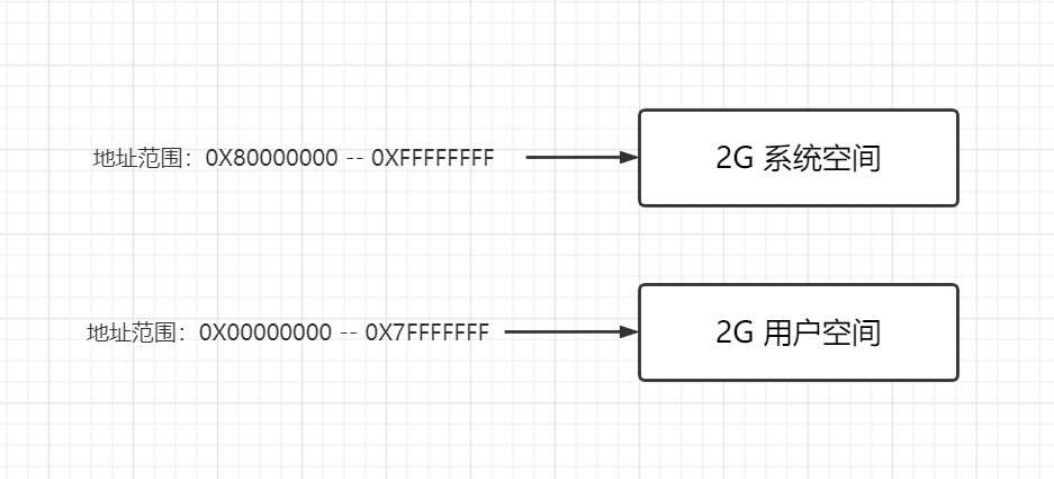


通过对比观察，可以看出两个项目的参数1的地址都为0x00400000（在进程中的首地址），两者不是同一块内存，修改一个，另一个的程序的参数1的地址不会进行修改。两者户不影响。

每启动一个程序都可以将其称为：进程。
进程和线程的关系，一个进程可以对应多个线程。

进程的内存分布（虚拟内存）：
32位程序，程序最多可访问4GB空间，每个进程都有4GB，高2G给系统，低2G给用

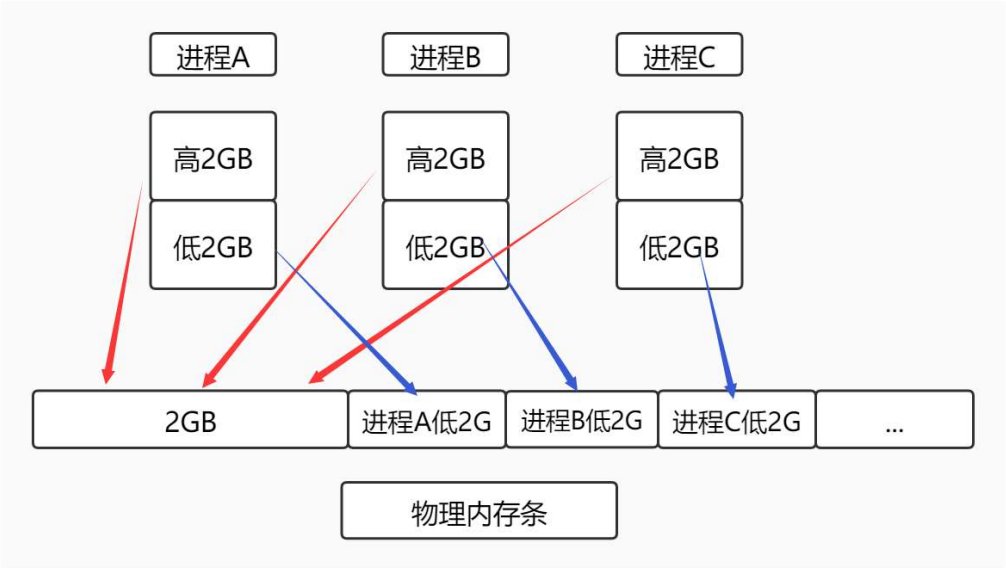
户（0~64k(10000)受保护）。



32位的CPU最大支持内存4GB，物理内存为4GB时，每个进程都有4GB，这点是怎样做到的?

进程间进行内存隔离：

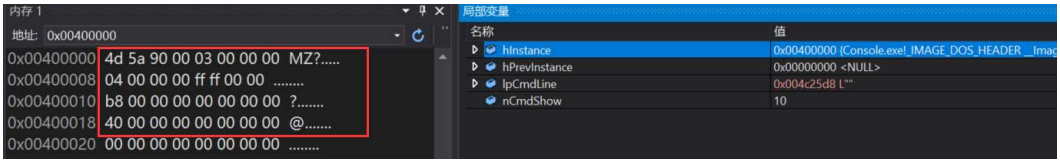
所有进程的高2G对应内存条的前2GB， 所有进程的低2G分别存储在物理内存条剩下的内存中，位置不会冲突，修改谁的就是修改谁的。



结论： 进程间的内存是相互隔离的。

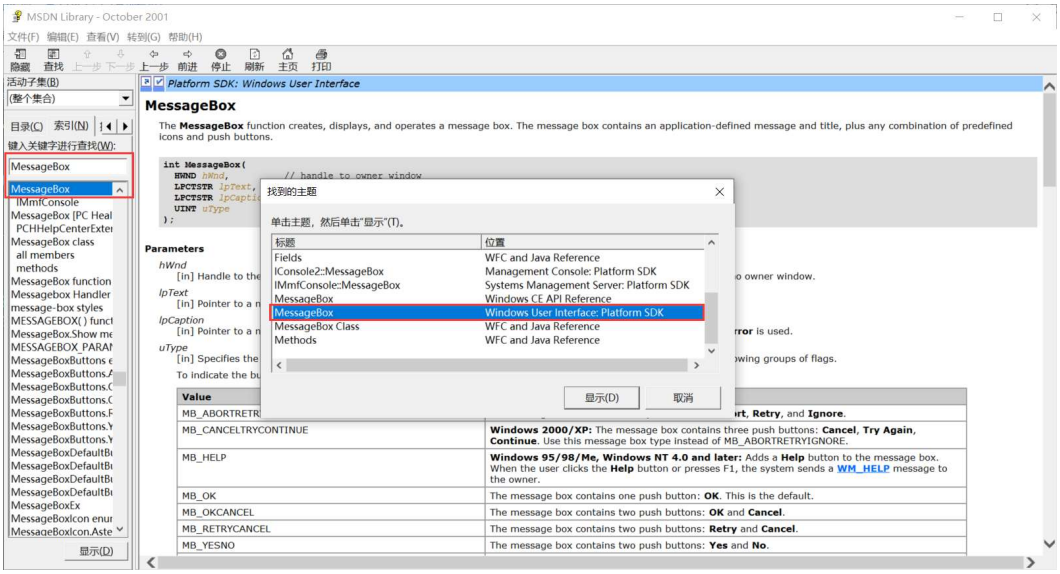
上述两个程序中的参数1的实例句柄的地址都为0x00400000，0x00400000这个地址其实就是一个程序或者一个进程在内存中的首地址，从这个地址开始，装载程序的时候，从这个地址往后面拷贝代码。

置于地址上的内容一样的原因是其存在固定格式的。



SDK版的HelloWorld

需要使用的函数：MessageBox



MessageBox 函数介绍

函数功能

MessageBox 函数用于显示一个模态对话框，其中包含一个系统图标，一组按钮和一个简短的特定于应用程序消息，如状态或者错误信息。

API 函数原型

注释：_In_ 说明该参数是输入的，_opt_ 说明该参数是可选参数。

```
int WINAPI MessageBox(
  _In_opt_  HWND hWnd,
  _In_opt_  LPCTSTR lpText,
  _In_opt_  LPCTSTR lpCaption,
  _In_      UINT uType
);
```

参数解析：

参数	含义
hWnd	要创建的消息框的所有者窗口的句柄。如果此参数为 NULL，则消息框没有所有者窗口
lpText	消息框的内容
lpCaption	消息框的标题

参数	含义
uType	1、指定一个决定对话框的内容和行为的位标志集 2、此参数可以通过指定下列标志或标志的组合，来显示消息框中的按钮以及图标

uType参数定义解析：

按钮	含义
MB_OK	默认值，有一个 "确定" 按钮在里面
MB_YESNO	有 "是" 和 "否" 两个按钮在里面
MB_ABORTRETRYIGNORE	有 "终止"，"重试" 和 "跳过" 三个按钮在里面
MB_YESNOCANCEL	有 "是"，"否" 和 "取消" 三个按钮在里面
MB_RETRYCANCEL	有 "重试" 和 "取消" 两个按钮在里面
MB_OKCANCEL	有 "确定" 和 "取消" 两个按钮在里面

图标	含义
MB_ICONEXCLAMATION	一个惊叹号出现在消息框： 
MB_ICONWARNING	一个惊叹号出现在消息框（同上）
MB_ICONINFORMATION	一个圆圈中小写字母i组成的图标出现在消息框： 
MB_ICONASTERISK	一个圆圈中小写字母i组成的图标出现在消息框（同上）
MB_ICONQUESTION	一个问题标记图标出现在消息框： 
MB_ICONSTOP	一个停止消息图标出现在消息框： 
MB_ICONERROR	一个停止消息图标出现在消息框（同上）
MB_ICONHAND	一个停止消息图标出现在消息框（同上）

默认按钮	含义
MB_DEFBUTTON1	指定第一个按钮为默认按钮
MB_DEFBUTTON2	指定第二个按钮为默认按钮
MB_DEFBUTTON3	指定第三个按钮为默认按钮
MB_DEFBUTTON4	指定第四个按钮为默认按钮

消息框形态	含义
MB_APPLMODAL	1. 在 hWnd 参数标识的窗口中继续工作以前，用户一定响应消息框 2. 但是，用户可以移动到其他线程的窗口且在这些窗口中工作 3. 根据应用程序中窗口的层次机构，用户则以移动到线程内的其他窗口 4. 所有母消息框的子窗口自动地失效，但是弹出窗口不是这样 5. 如果既没有指定 MB_SYSTEMMODAL 也没有指定 MB_TASKMODAL，则 MB_APPLMODAL 为默认的
MB_SYSTEMMODAL	1. 除了消息框有 WB_EX_TOPMOST 类型，否则 MB_APPLMODAL 和 MB_SYSTEMMODAL 一样 2. 用系统模态消息框来改变各种各样的用户，主要的损坏错误需要立即注意（例如，内存溢出） 3. 如果不是那些与 hWnd 联系的窗口，此标志对用户对窗口的相互联系没有影响
MB_TASKMODAL	1. 如果参数 hWnd 为 NULL 的话，那么除了所有属于当前线程高层次的窗口失效外，MB_TASKMODAL 和 MB_APPLMODAL 一样 2. 当调用应用程序或库没有一个可以得到的窗口句柄时，可以使用此标志，但仍需要阻止输入到调用线程的其他窗口，而不是搁置其他线程

其他标志	含义
MB_DEFAULT_DESKTOP_ONLY	1. 接收输入的当前桌面一定是一个默认桌面，否则函数调用失败 2. 默认桌面是一个在用户已经记录且以后应用程序在此上面运行的桌面
MB_HELP	1. 把一个 Help 按钮增加到消息框 2. 选择 Help 按钮或按 F1 产生一个 Help 事件
MB_RIGHT	文本为右对齐
MB_RTLREADING	用在 Hebrew 和 Arabic 系统中从右到左的顺序显示消息和大写文本
MB_SETFOREGROUND	1. 消息框变为前景窗口 2. 在内部系统为消息个调用 SetForegroundWindow 函数
MB_TOPMOST	消息框用 WS_EX_TOPMOST 窗口类型来创建 MB_SERVICE_NOTIFICATION

返回值：

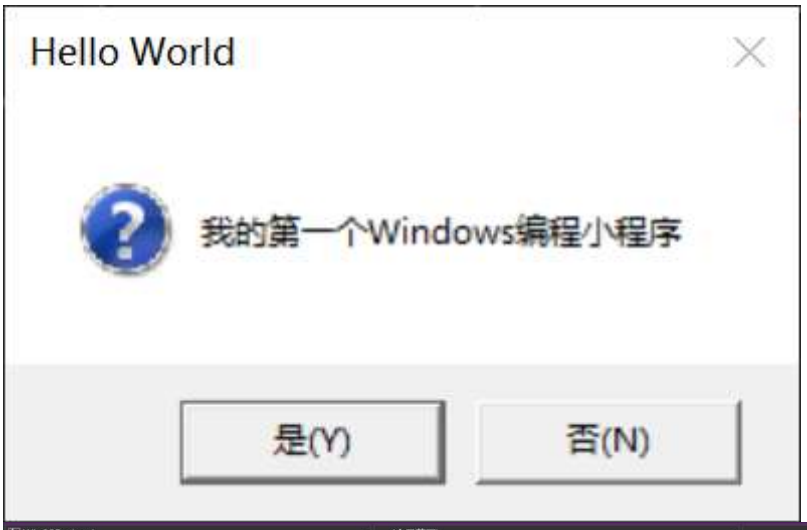
返回值	含义
IDOK	用户按下了"确认"按钮
IDCANCEL	用户按下了"取消"按钮
IDABORT	用户按下了"中止"按钮
IDRETRY	用户按下了"重试"按钮
IDIGNORE	用户按下了"忽略"按钮
IDYES	用户按下了"是"按钮
IDNO	用户按下了"否"按钮

代码示例：

```
#include <Windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR
szCmdLine, int iCmdShow)
{
        MessageBox(NULL, TEXT("我的第一个Windows编程小程序"),
TEXT("Hello World"), MB_YESNO | MB_ICONQUESTION | MB_DEFBUTTON1);
        return 0;
}
```

程序运行效果图：




```
8625  #define MB_CANCELTRYCONTINUE      0x00000006L
8626  #endif /* WINVER >= 0x0500 */
8627
8628
8629  #define MB_ICONHAND               0x00000010L
8630  #define MB_ICONQUESTION           0x00000020L
8631  #define MB_ICONEXCLAMATION        0x00000030L
8632  #define MB_ICONASTERISK           0x00000040L
8633
8634  #if(WINVER >= 0x0400)
8635  #define MB_USERICON               0x00000080L
8636  #define MB_ICONWARNING            MB_ICONEXCLAMATION
8637  #define MB_ICONERROR              MB_ICONHA
8638  #endif /* WINVER >= 0x0400 */
8639
8640  #define MB_ICONINFORMATION        MB_ICONASTERISK
8641  #define MB_ICONSTOP               MB_ICONHAND
8642
8643  #define MB_DEFBUTTON1             0x00000000L
8644  #define MB_DEFBUTTON2             0x00000100L
8645  #define MB_DEFBUTTON3             0x00000200L
8646  #if(WINVER >= 0x0400)
8647  #define MB_DEFBUTTON4             0x00000300L
```

#define MB_ICONEXCLAMATION 0x00000030L

可以进行或运算

> 参考文章MessageBox

MessageBox 提供两个版本，MessageBoxA和MessageBoxW（Ascii 和 Unicode）

```
8688  WINAPI
8689  MessageBoxA(
8690      _In_opt_ HWND hWnd,
8691      _In_opt_ LPCSTR lpText,
8692      _In_opt_ LPCSTR lpCaption,
8693      _In_ UINT uType);
8694  WINUSERAPI
8695  int
8696  WINAPI
8697  MessageBoxW(
8698      _In_opt_ HWND hWnd,
8699      _In_opt_ LPCWSTR lpText,
8700      _In_opt_ LPCWSTR lpCaption,
8701      _In_ UINT uType);
8702  #ifdef UNICODE
8703  #define MessageBox  MessageBoxW
8704  #else
8705  #define MessageBox  MessageBoxA
8706  #endif // !UNICODE
```

参数名称相同，但是参数的类型不同

参数的类型

代码示例：

```
#include <Windows.h>
#include <tchar.h>
```

```
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
    _In_opt_ HINSTANCE hPrevInstance,
    _In_ LPWSTR lpCmdLine,
    _In_ int nCmdShow)
{
    MessageBox(NULL,
        _T("我的第一个SDK小程序"),
        _T("这是一个标题"),
    MB_YESNOCANCEL);
    // _T 兼容 Ascii 和 Unicode

    // Windows提供了一个类似 _T的宏 TEXT
    MessageBox(NULL,
        TEXT("我的第一个SDK小程序"),
        TEXT("这是一个标题"),
        MB_YESNOCANCEL);
    // WindowsSDk 不跨平台
    return 0;
}
```

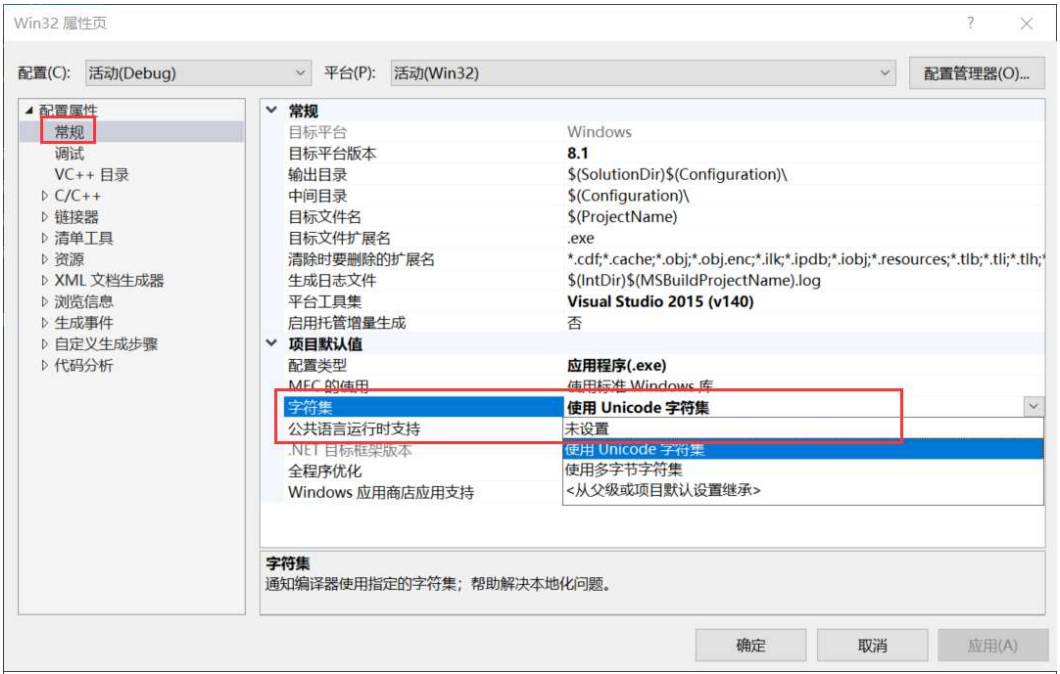
	头文件	开关	兼容宏
C	tchar.h	_UNICODE	_T
Windows	Windows.h	UNICODE	TEXT

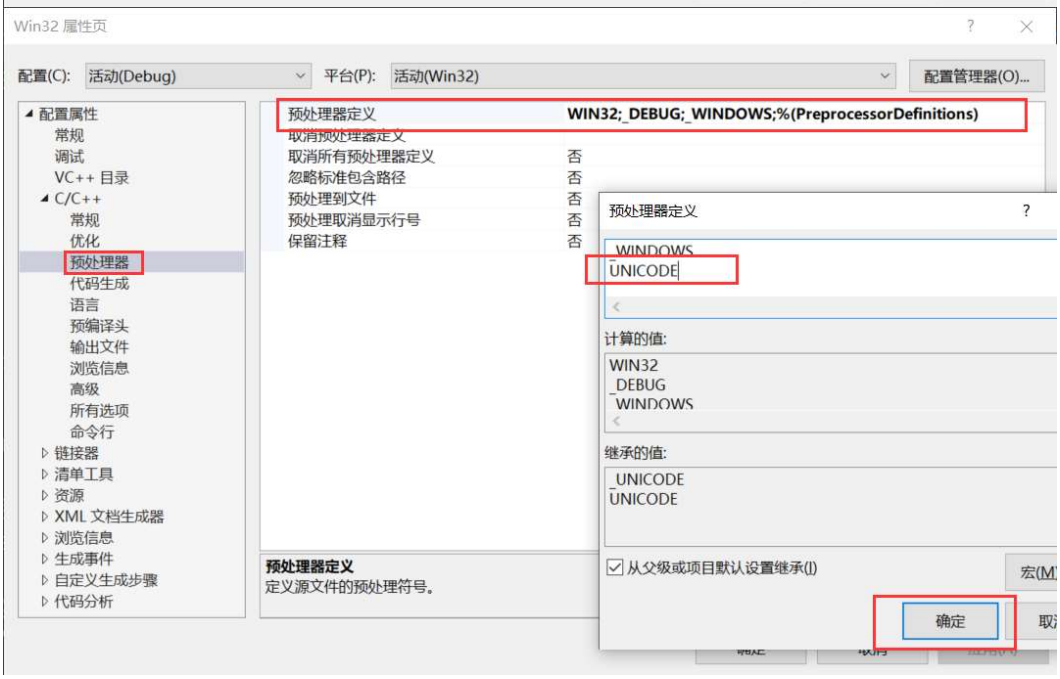
宏的三个使用位置

- 使用#define UNICODE(W版本)，在头文件之前定义
- 编译选项 -- 项目属性 -- C/C++ -- 预处理器进行添加
- 编译选项 -- 项目属性 -- 常规 -- 字符集

操作如下图所示：

```
7   #define UNICODE
8
9   int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
```





MessageBox 参数类型拆分分析：

```
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
    _In_opt_ HINSTANCE hPrevInstance,
    _In_ LPWSTR lpCmdLine,
    _In_ int nCmdShow)
{
    /*
    MessageBox(NULL,
        _T("我的第一个SDK小程序"),
        _T("这是一个标题"),
        MB_YESNOCANCEL);
    */
    // _T 兼容 Ascii 和 Unicode
```

```
    // Windows提供了一个类似 _T的宏 TEXT
    MessageBox(NULL,
        TEXT("我的第一个SDK小程序"),
        TEXT("这是一个标题"),
        MB_YESNOCANCEL);
    // WindowsSDk 不跨平台

    //HWND
    //DWORD // unsigned long

    /*
    HWND hWndTest
    HINSTANCE hInstance --> 这样写方便日后程序升级，可读性要比下面的要好
    void* pInstance
    vois* pWnd
    */

    // UINT -- unsigned int
    // HINSTANCE --> H INSTANCE --> handle instance
    // HWND --> H WND --> handle window
    // LPWSTR --> LP W STR -->
    // long pointer wide str 32位CPU为了和16位进行区分，兼容16位
    return 0;
}
```