孙鑫MFC精简难点笔记

相关代码文件存放位置：E:\software\c\D\MFC\VS2012\Sunxin

§Lesson1\_Win32SDK

1、

int WINAPI WinMain(        
  
    HINSTANCE *hInstance*,

    HINSTANCE *hPrevInstance*,

    LPSTR *lpCmdLine*,

    int *nCmdShow*

);

第三个参数lpCmdLine在VC6.0中的设置位置在：Project->Setting->Debug->Program arguments

2、什么是层叠窗口

在CreateWindows中WS\_OVERLAPPED这个窗口类型代表的就是层叠窗口，层叠窗口说得是带有标题栏和边框的窗口。

3、\_\_stdcall与\_\_cdecl

\_\_stdcall是标准调用约定，\_\_cdecl是c语言调用约定，除了变参的函数，其他API函数都是\_\_stdcall的调用习惯，但是VC++程序缺省默认情况下是我们的\_\_cdecl，所以需要加上\_\_stdcall的调用修饰符以便使用标准调用约定。

可以在此处更改调用约定：Project->Setting->C/C++->Code Generation(代码生成)->Calling convention(调用约定)

§Lesson3\_MFC

4、在MFC那五个类(CMainFrame CTestView CTestApp CTestDoc CAboutDlg )中，没有WinMain函数，是因为WinMain函数在linker链接的时候链接到我们的程序当中。

5、MFC运行脉络

APPMODULE.CPP：\_tWinMain所在地(#define \_tWinMain WinMain)，在\_tWinMain函数中调用了框架函数AfxWinMain

// 唯一的一个 CLesson3\_MFCApp 对象

//全局对象，代表这个应用程序本身。

CLesson3\_MFCApp theApp;

因为是全局对象，所以会先构造theApp这个对象，然后调用该类的构造函数，在这之前，需要先调用基类CWinApp的构造函数，在MFC源文件中搜一下CWinApp的相关文件。

APPCORE.CPP：和CLesson3\_MFCApp(继承自CWinApp)的全局对象theApp有关系。

WINMAIN.CPP：框架函数AfxWinMain所在地，在这里完成了设计窗口类、注册窗口类、创建窗口、显示和更新窗口等窗口相关的操作，还有消息循环。里面有可以通过AfxGetApp()获得指向CLesson3\_MFCApp的指针，调用了InitInstance，因为InitInstance是虚函数，所以调用了子类的InitInstance，即CLesson3\_MFCApp的InitInstance。在InitInstance中完成了窗口的相关工作，消息循环在Run函数中完成。

WINCORE.CPP：里面有详细的注册窗口类、创建窗口、显示和更新窗口等窗口相关的操作函数。CMainFrm中的PreCreateWindow在窗口创建完成之前多次被调用，可以在这里修改窗口的表现形式。

6、框架窗口和视类窗口、客户区和非客户区

框架窗口就是应用程序窗口，视类窗口是除去标题栏、菜单栏、工具栏之外的窗口；

框架窗口的客户区是除去标题栏、菜单栏的，注意，工具栏所处位置是客户区，框架窗口的非客户区包括标题栏、菜单栏。

视类窗口的客户区是除去标题栏、菜单栏、工具栏的，也就是说视类窗口没有非客户区。

7、窗口创建时消息发送次序：先WM\_CREATE(CreateWindow)，然后WM\_SIZE和WM\_SHOWWINDOW(ShowWindow),最后WM\_PAINT(UpdateWindow)。

8、CButton也是一个窗口，也需要创建(m\_btn.Create)，也需要显示(m\_btn.ShowWindow(SW\_SHOWNORMAL))，如果在窗口类型中加入了WS\_VISIBLE，就不需要ShowWindow了。Cbutton类对象同样也有DestroyWindow。

§Lesson4\_Draw

9、Simple value types:

CPoint CSize CRect CString CTime

CDC :CClientDC CWindowDC CPaintDC CMetaFileDC

CGdiObject:CPen CBrush CFont CBitmap CPalette CRgn

10、View类始终覆盖在框架类之上的

11、获取桌面窗口的句柄：**GetDesktopWindow**

12、SetROP2 光栅二元操作要画的像素点和已知像素点进行互操作

§Lesson5\_Text

13、SelectClipPath 通过BeginPath和EndPath来创建一个路径层，类似于西方国家在古代的时候圈地运动，圈的一块地为自己的，其他人不可使用。路径层和已知的剪切区域（Clipping Region）进行互操作

14、**GetDeviceCaps**返回设备属性，比如，以像素为单位返回显示器的宽度和高度

**GetSystemMetrics**得到系统相关属性

**GetTextMetrics**得到文本相关属性比如文本的字符高度文本的平均字符宽度和最大字符宽度

**GetTextExtent**得到函数参数中文本的实际长度，返回一个CSize对象。

15、用dc.DrawText来实现卡拉OK的功能

§Lesson6\_Menu

17、新建一个菜单项，通过在MFC四个类（CMenuView CMenDoc CMainFrame CMenuApp）分别响应这个菜单项的方式来确定菜单的WM\_COMMAND消息在四个类之间的路由情况，得出结论是：CMenuView ->CMenDoc -> CMainFrame ->CMenuApp，当有一个类响应了命令消息之后，其他类不再对其响应。

18、命令消息的路由



首先，CMainFrame接受到消息，然后传递给CMenuView，查看CMenuView是否对命令消息进行了响应，如果CMenuView没有对消息进行响应，CMenuView就把消息传递给CMenuDoc，查看CMenuDoc是否对消息进行了响应，如果CMenuDoc也没有对消息进行响应，则把消息传递给CMenuView，CMenuView会把消息返回给CMainFrame，查看CMainFrame是否对消息进行了响应，如果CMainFrame没有对消息进行响应，则把消息传递给CMenuApp

，查看CMenuApp是否对消息进行了响应。

对标准消息（WM\_TIMER）的响应在OnWndMsg函数进行，对命令消息（包括静态菜单和动态菜单、和右键弹出式菜单）的响应在OnCommand函数里面进行，对通告消息的响应在OnNotify函数里面进行。可以通过覆盖CMainFrame里面的虚函数OnCommand函数来阻止命令消息的路由。

19、菜单的结构：菜单栏GetMenu->子菜单GetSubMenu->菜单项

20、EnableMenuItem的使用注意事项

// The code fragment below shows how to disable (and gray out) the

// File\New menu item.

// NOTE: m\_bAutoMenuEnable is set to FALSE in the constructor of

// CMainFrame so no ON\_UPDATE\_COMMAND\_UI or ON\_COMMAND handlers are

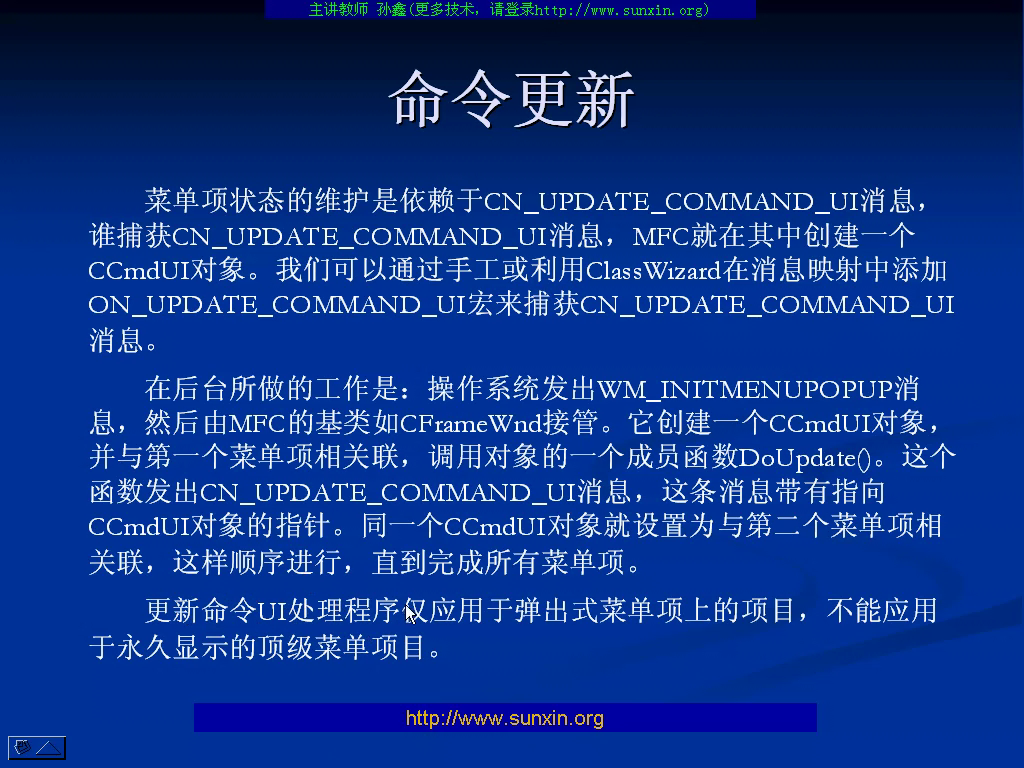
// needed, and CMenu::EnableMenuItem() will work as expected.

CMenu\* mmenu = GetMenu();

CMenu\* submenu = mmenu->GetSubMenu(0);

submenu->EnableMenuItem(ID\_FILE\_NEW, MF\_BYCOMMAND | MF\_DISABLED | MF\_GRAYED);//一般来说，不可用需和灰色标志同时使用

21、菜单项的命令更新机制



22、菜单项和工具栏的索引和ID号，利用ID号可同时标识菜单项和工具栏，但是利用索引并不一定能索引到工具栏。为了让菜单项和工具栏上的图标保持一致，建议使用ID号。

23、添加(AppendMenu）、插入(InsertMenu)、删除菜单(DeleteMenu)（包括弹出菜单POPUP MENU和菜单项），用MF\_POPUP、MF\_STRING和MF\_SEPARATOR等标志。

24、PhoneBook，VS2012版，见程序文件夹。

25、

父窗口(CMainFrm)获取当前活动的子窗口(CTestView)的指针：GetActiveView，请仔细阅读MSDN上关于该函数的使用，在多文档结构中，不能使用该函数，因为多文档中，该视类不属于任何其中一个文档。

CTestView获取CMainFrm的一个指针：GetParent，注意它与GetParentOwner的区别。Owner是最顶层的那个，它不是child类型的。

CTestDoc获取CTestView的一个指针：先用GetFirstViewPosition来获取视类列表中第一个视类对象的位置（PS:对于一个文档类对象来说，可以有多个视类对象与之相关，但对于一个视类对象来说，只能有一个文档对象与之相关。），然后用GetNextView来得到视类对象的指针。

获取CTestDoc指针：GetDocument

可以在其他类文件中用extern CTestApp theApp，来使用CTestApp的对象。

26、CObjectArray中的CStringArray和CPtrArray的详细使用。

§Lesson7\_Dialog

27、

DDX\_ 和DDV\_

Dialog Data Exchange/Validate

28、已知ID号获取句柄GetDlgItem 已知句柄获取ID号GetDlgCtrlID

29、三大类操控对话框上面的控件的方法：

（1）GetDlgItem、GetWindowText和SetWindowText。

（2）如果采用控件关联变量方式来操作，需要使用UpdateData来初始化和更新值。比较经典的例子就是UpdataData();/\*获取对话框上面的值\*/m\_num3 = m\_num1 + m\_num2;UpdataData(FALSE)//更新值到对话框上

（3）发送消息的方法，消息是WM\_GETTEXT和WM\_SETTEXT，采用SendMessage或者SendDlgItemMessage来发送。

30、

WM\_(Windows Message)和EM\_(Edit Message)、UM(User Message)、WM\_USER

ON\_WM\_TIMER()、ON\_COMMAND()、ON\_UPDATE\_COMMAND\_UI()、ON\_MESSAGE()

31、

GetWindowRect、GetClientRect和SetWindowPos、MoveWindow

32、

SetClassLong、GetClassLong和SetWindowLong、GetWindowLong

33、

**GetNextDlgTabItem**(GetFocus())->SetFocus();

§Lesson8\_CPropertyPage

34、CPropertySheet和CPropertyPage

CPropertySheet继承自CWnd，而CPropertyPage继承自CDialog，其操作与CDialog类似。

35、CPropertyPage里面的虚函数：

OnOk OnCancel

OnSetActive OnKillActive

OnWizardBack OnWizardNext OnWizardFinish

§Lesson9\_Style

36、window styles

MSDN中搜索window styles，然后The SDI Case部分：

In a single document interface (SDI) application, the default window style in the framework is a combination of the WS\_OVERLAPPEDWINDOW and FWS\_ADDTOTITLE styles. FWS\_ADDTOTITLE is an MFC-specific style that instructs the framework to add the document title to the window’s caption. To change the window attributes in an SDI application, override the PreCreateWindow function in your class derived from CFrameWnd (which AppWizard names CMainFrame). For example:

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)

{

   // Create a window without min/max buttons or sizable border

   cs.style = WS\_OVERLAPPED | WS\_SYSMENU | WS\_BORDER;

   // Size the window to 1/3 screen size and center it

   cs.cy = ::GetSystemMetrics(SM\_CYSCREEN) / 3;

   cs.cx = ::GetSystemMetrics(SM\_CXSCREEN) / 3;

   cs.y = ((cs.cy \* 3) - cs.cy) / 2;

   cs.x = ((cs.cx \* 3) - cs.cx) / 2;

   // Call the base-class version

   return CFrameWnd::PreCreateWindow(cs);

}

This code creates a main frame window without Minimize and Maximize buttons and without a sizable border. The window is initially centered on the screen.

37、获取当前应用程序的实例句柄

AfxGetInstanceHandle：获取当前实例的句柄，等价于AfxGetApp()->m\_hInstance

38、

AfxRegisterWndClass与AfxRegisterClass

39、

// TODO: 如果不需要可停靠工具栏，则删除这三行

m\_wndToolBar.EnableDocking(CBRS\_ALIGN\_ANY);//工具栏可以停靠

EnableDocking(CBRS\_ALIGN\_ANY);//可以被停靠

DockControlBar(&m\_wndToolBar);

菜单栏更改之后记得要用**DrawMenuBar**来重绘，工具栏更改之后记得要用**RecalcLayout**来重排工具栏，重排完成之后还要DockControlBar将工具栏停靠。

40、

隐式类型转换有两种：一种是构造函数，一种是重载。

41、CGdiObject::DeleteObject

Deletes the attached Windows GDI object from memory by freeing all system storage associated with the Windows GDI object. The storage associated with the CGdiObject object is not affected by this call. An application should not call DeleteObject on a CGdiObject object that is currently selected into a device context.

42、IsWindowVisible、IsWindowEnabled

43、CStatusBar设置状态栏的文本的三种不同方式：

Call [CWnd::SetWindowText](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN\2001OCT\1033\vcmfc.chm::/html/_mfc_cwnd.3a3a.setwindowtext.htm) to update the text in pane 0 only.

Call [CCmdUI::SetText](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN\2001OCT\1033\vcmfc.chm::/html/_mfc_ccmdui.3a3a.settext.htm) in the status bar’s ON\_UPDATE\_COMMAND\_UI handler.

Call [SetPaneText](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN\2001OCT\1033\vcmfc.chm::/html/_mfc_cstatusbar.3a3a.setpanetext.htm) to update the text for any pane.

44、进度栏CprogressCtrl

45、新建工具栏的完美模拟

static CRect rectPrev;

static bool isFloat = false;

if(m\_newToolBar.IsWindowVisible()){

if(m\_newToolBar.**IsFloating**()){

isFloat = true;

//::GetWindowRect(m\_newToolBar, &rectPrev);//这句获取的坐标不行，得用下////面这句

m\_newToolBar.**GetDockingFrame()**->GetWindowRect(&rectPrev);

/\*上面代码中的GetDockingFrame我刚开始看不明白，我觉得这样是返回的主框架窗口的对象，怎么用它调用GetWindowRect获取的是工具栏的位置呢？后来看到下面片段(出处：VC++编程技术与难点剖析)

"同时，CControlBar还定义了成员函数GetDockingFrame()，取得当前的所有者框架窗口（主框架窗口或为浮动弹出的简化框架窗口）"

原来当工具栏以浮动独立窗口显示时它就有独立的简化框架窗口了，所以GetDockingFrame的：调用该成员函数获取指向您的控制条停靠的当前框架窗口。获取的当前框架窗口就是这个弹出的浮动简化框架窗口，而非主框架窗口

-------------------------------------

CFrameWnd::ShowControlBar这个函数直接显示和隐藏，并在原位置 \*/

m\_newToolBar.ShowWindow(SW\_HIDE);

DockControlBar(&m\_newToolBar);

RecalcLayout();

return;

}

m\_newToolBar.ShowWindow(SW\_HIDE);

isFloat = false;

}

else{

m\_newToolBar.ShowWindow(SW\_SHOWNORMAL);

}

DockControlBar(&m\_newToolBar);

RecalcLayout();

if(isFloat){

m\_newToolBar.ShowWindow(SW\_SHOWNORMAL);

**FloatControlBar**(&m\_newToolBar, CPoint(rectPrev.left,rectPrev.top));

return;

}

其实，以上这些代码可以用一句代码代替：

**ShowControlBar**(&m\_newToolBar,!m\_newToolBar.IsWindowVisible(),FALSE);

§Lesson10\_Graphic

45、

CColorDialog CFontDialog CFileDialog

46、窗口类和窗口，其中关系的纽带就是m\_hWnd(资源的句柄)。

CGDIObject与Windows GDI Object，其中关系的纽带就是m\_hObject(资源的句柄)。

47、

关于对话框控件的相关操作（比如自绘）请看孙鑫老师第10集的1小时30分钟左右的视频

SetTextColor、SetBkMode、SetBkColor

在Windows SDK中，没有WM\_CTLCOLOR消息，只有WM\_CTLCOLORBTN WM\_CTLCOLORSTATIC这类消息。

在MFC中，有WM\_CTLCOLOR这个消息，里面处理各种消息：

CTLCOLOR\_BTN   Button control

CTLCOLOR\_DLG   Dialog box

CTLCOLOR\_EDIT   Edit control

CTLCOLOR\_LISTBOX   List-box control

CTLCOLOR\_MSGBOX   Message box

CTLCOLOR\_SCROLLBAR   Scroll-bar control

CTLCOLOR\_STATIC   Static control

48、**WM\_DRAWITEM**

**CButton::DrawItem**

virtual void DrawItem( LPDRAWITEMSTRUCT lpDrawItemStruct );

Parameters

lpDrawItemStruct

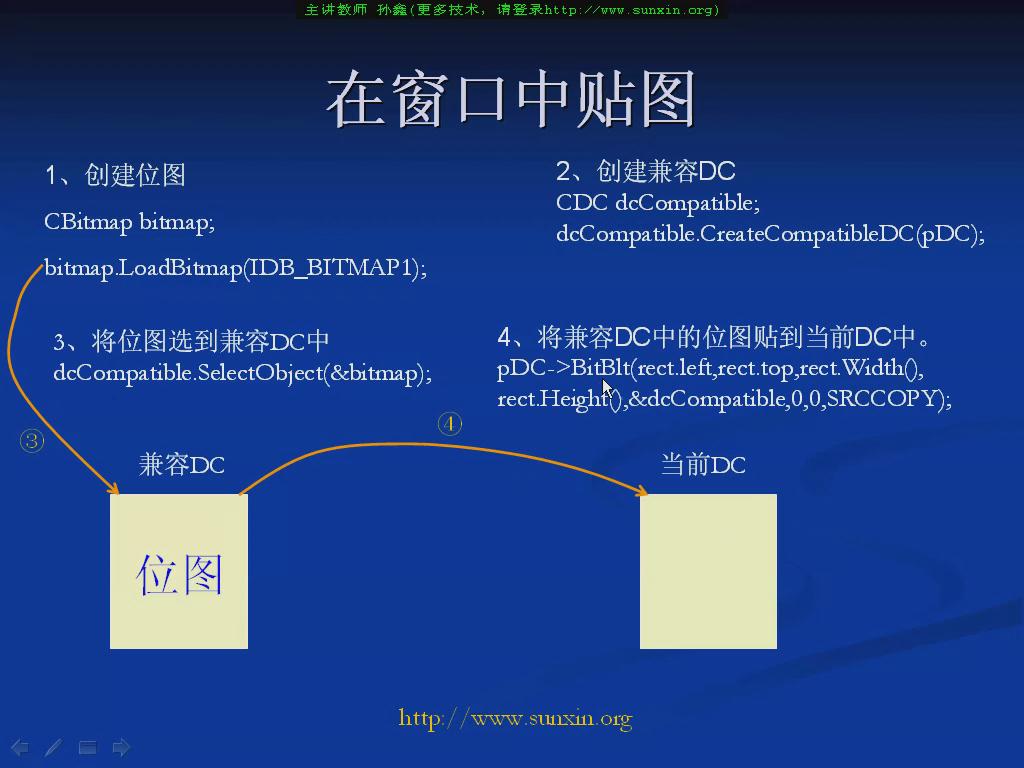
A long pointer to a [DRAWITEMSTRUCT](mk:@MSITStore:C:\Program%20Files\Microsoft%20Visual%20Studio\MSDN\2001OCT\1033\vcmfc.chm::/html/_mfc_drawitemstruct_structure.htm) structure. The structure contains information about the item to be drawn and the type of drawing required.

Remarks

Called by the framework when a visual aspect of an owner-drawn button has changed. An owner-drawn button has the BS\_OWNERDRAW style set. Override this member function to implement drawing for an owner-drawn CButton object. The application should restore all graphics device interface (GDI) objects selected for the display context supplied in lpDrawItemStruct before the member function terminates.

49、两个Button类：一个是孙鑫写的CSXBtn，一个是网上的人写的CButtonST。

50、在窗口中贴图（通过创建兼容DC）



第二步中：是创建相对于当前DC兼容的一个DC，即兼容DC。

WM\_ERASEBKGND（最好是在这个消息里面贴图，闪烁较小） WM\_PAINT(也可在这个消息内贴图，但是闪烁较大)

§Lesson11\_Graphic

图形的保存和重绘

51、

我们绘制的图形要想在窗口中始终显示，需要保存。

每次窗口重绘都会发送一个WM\_PAINT消息，OnPaint函数会被调用，View类中的OnDraw函数也就会被调用，那么，如何保存。

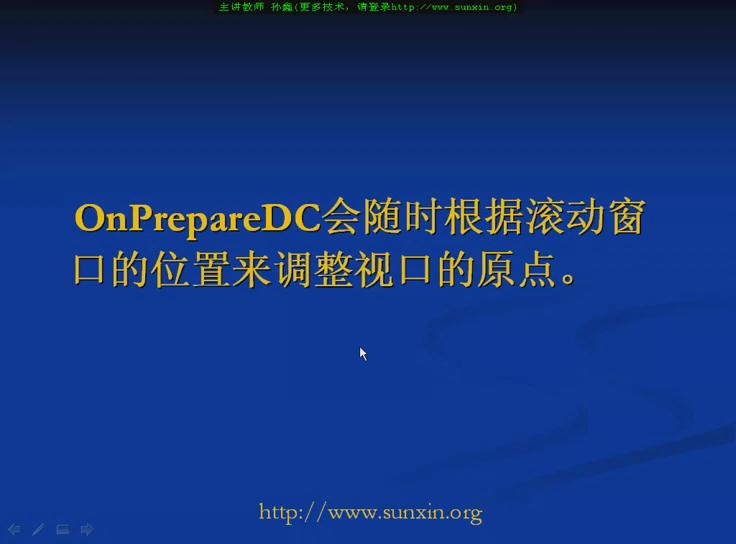
52、SetScrollSizes第一个参数需要设置映射模式，**SetMapMode**用来设置映射模式。

53、基本上所有的GDI函数（不包括GetDeviceCaps）都使用逻辑坐标，因为他们可以在设备坐标之外绘图而不仅仅局限于客户区，而所有的消息（WM\_SIZE、WM\_MOUSEMOVE）都使用设备坐标，因为他们触发窗口发生消息响应的时候必须处于窗口中才能触发。

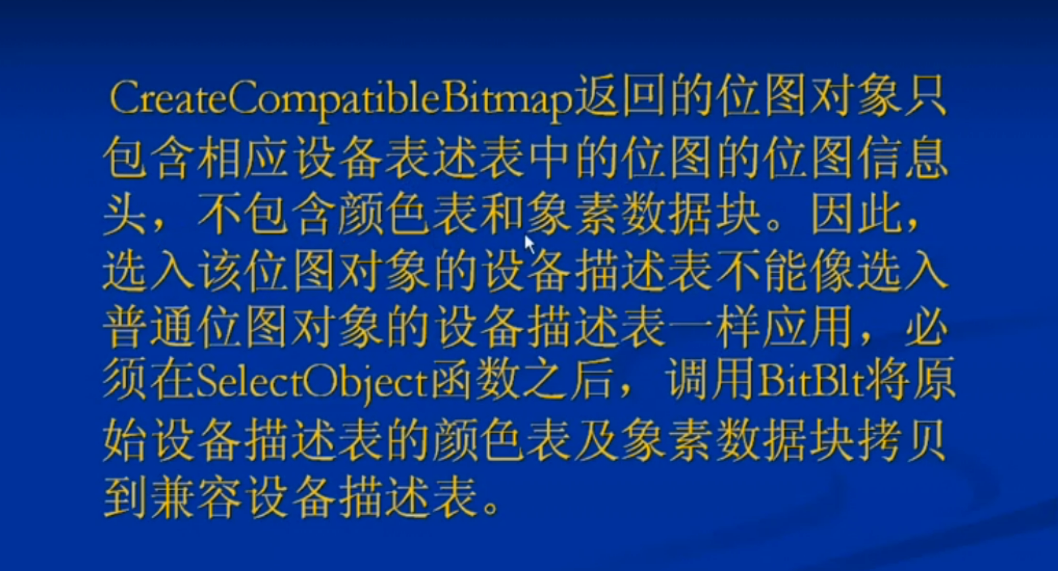
54、**OnInitialUpdate**函数是在窗口完全创建之后第一个调用的函数，它还在OnDraw函数调用之前。

55、DPtoLP（设备坐标转化为逻辑坐标）与LPtoDP（逻辑坐标转化为设备坐标）

56、



57、用CMetaFileDC对象来创建元文件，然后来播放和删除该元文件。还有另外一种方法来保存图形，利用兼容DC，兼容位图，通过把兼容位图选到兼容DC里面来确定兼容DC显示表面的大小。



§Lesson12\_File

58、C++读写文件

写入数据 输出流 相对于流来说是输出 输出到文件 ofstream

读出数据 输入流 相对于流来说是输入 输入到流 ifstream

#include <fstream.h>

ofstream ofs(“4.txt”);

ofs.write(“<http://www.sunxin.org>”, strlen(“<http://www.sunxin.org>”));

ofs.close();

ifstream ifs(“4.txt”);

char ch[100];

memset(ch, 0, 100);

ifs.read(ch, 100);

ifs.close()’

MessageBox(ch);

59、Windows API文件读写

**CreateFile WriteFile ReadFile**

CreateFile函数中的dwFlagsAndAttributes参数中有一个标记FILE\_FLAG\_OVERLAPPED，它的工作方式是异步I/O方式，也就是层叠I/O方式，函数立即返回，操作系统会使用线程来完成这个I/O操作，如果没有加上这个标记，采用的方式是同步I/O方式。

60、INI文件和注册表的一些操作RegSetValue等等

§Lesson13\_CArchive

文档与串行化

61、在30分钟左右的讲述了文档类、框架类、视类如何通过文档管理器(CDocManger)、文档模板（单文档模板）来三位一体（三码合一）的。

62、在程序开始建立时（程序建立时需要建立一个单文档，文档的名称叫无标题）和单击文件新建命令时，会调用**OnNewDocument**这个虚函数。可以在OnNewDocument函数里面通过SetTitle来设置文档的标题。

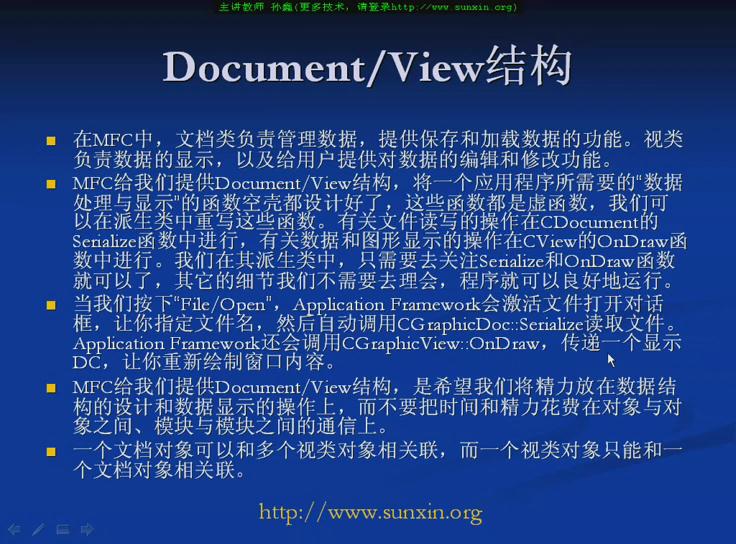
单击文件打开或保存命令时，会调用Doc类中的**Serialize**这个函数来加载或保存数据。

63、在单文档程序中，不能对该文档保存了之后，马上就去打开，因为MFC检查到了当前的文档就是你要打开的文档，所以就不会再去打开，和word很类似。

64、**CDocTemplate::GetDocString**

65、这是1小时之后的内容：要用CArchive这个类来保存一个对象，那么这个对象它的类必须支持串行化。**CObArray** incorporates the **IMPLEMENT\_SERIAL** macro to support serialization and dumping of its elements.[2014/4/26:串行化是不可继承的，比如第一个挣钱的程序给小橘子做的里面CDraw派生出来的CDrawLine等类均需要手动使之序列化]

m\_obArray.Serialize(ar);



66、**DeleteContents**

OnNewDocument和OnOpenDocument都会调用这个函数来删除文档数据但不会销毁CDocument对象。

OnSaveDocument函数

void CGraphicDoc::DeleteContents()

{

int iCount;

iCount = m\_obArray.GetSize();

while(nCount--){

delete m\_obArray.GetAt(nCount);

m\_obArray.RemoveAt(nCount);//这个地方还可以在while循环外用//m\_obArray.RemoveAll();

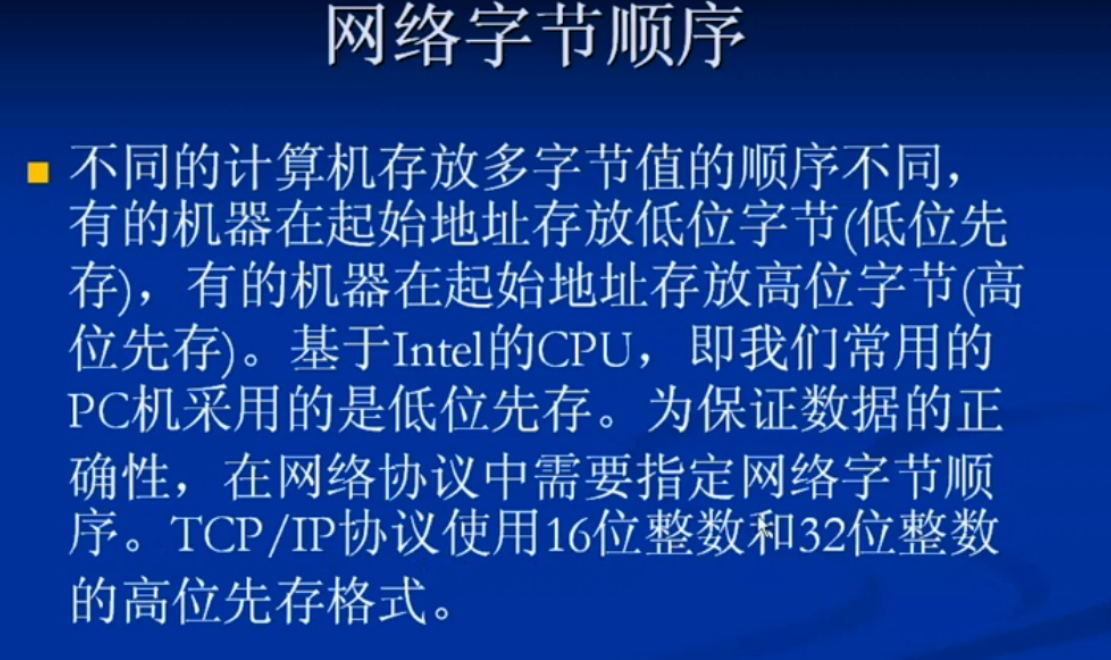
}

}

§Lesson14\_Net

网络编程

67、**网络字节序**



68、

我们基于TCP、UDP不同的网络协议程序，它们可以有相同的端口号。

69、

套接字类型：

基于TCP的套接字类型SOCK\_STREAM（字节流）

基于UDP的套接字类型SOCK\_DGRAM（datagrams）

原始套接字类型SOCK\_RAW



因为我们要在Server端接收Client的请求，所以我们要指示在Server机器上的哪个IP地址（哪个物理网卡）和哪个端口上接收Client的请求。所以用到了bind。

而在accept的时候，Server端就保存了Client的IP和端口信息。



基于UDP的socket，对C/S的这种概念不是很强化，我们可以把服务器端叫作接受端，发送数据的这一端（客户端）叫发送端。

70、

如果需要看代码，找不到的话，可以看第一次学习孙鑫教程写的代码。

地址在E:\software\c\D\MFC\sunxin，有两个10课的Lesson文件夹

71、

WSAStartup中WSA，一种解释是说Windows Socket API,一种解释是说Windows Socket Asynchronous，用来区分同步和异步。（Windows Socket Asynchronous 的缩写，传统的Socket都是同步阻塞的，如果要实现异步则要编程对socket I/O进行控制。  
Windows实现socketSDK，尤其是在MFC下是通过消息机制的，异步是指Socket I/O和主程序是不同步执行。所以有CSocket和CAsyncSocket之分。），个人比较支持异步的说法（2014/3/28）。

（2014/3/29）更新：但是当后来学到WSAAsyncSelect函数的时候，发现WSA中的A应该是第一种解释了。因为后面有Async了。



孙鑫好像MAKEWORD(x,y)说反了：

The MAKEWORD macro creates a WORD value by concatenating the specified values.

WORD MAKEWORD(

BYTE bLow,

BYTE bHigh

);

Parameters

bLow

Specifies the low-order byte of the new value.

bHigh

Specifies the high-order byte of the new value.

摘自MSDN2001

MSDN2001上还有说：The WSAStartup function must be the first Windows Sockets function called by an application or DLL.

在最后，我们都需要调用WSACleanup来终止winsock库的使用，在终止之前还可以用closesocket来关闭某个套接字。

72、

htonl htons (l:long s:short h:host n:network)

inet\_addr(“127.0.0.1”);

inet\_ntoa(“addrClient.sin\_addr”);(n:network(网络字节序) address:a string in Internet standard dotted format)

§Lesson15\_MultiThread

73、

进程由两个部分组成：

1、操作系统用来管理进程的**内核对象**。**内核对象**也是系统用来存放关于进程的统计信息的地方。

2、地址空间。它包含所有可执行模块或DLL模块的代码和数据。它还包含动态内存分配的空间。如线程堆栈和堆分配空间。

74、

线程有两部分：内核对象（比如互斥对象Mutex、事件对象Event），这是一类数据结构；地址空间（比如代码块、数据块）。

75、

进程是不活泼的。进程从来不执行任何东西，它只是线程的容器，或者说是线程的执行环境。进程所自动创建的第一个线程叫作主线程。此后，主线程可以创建其他的线程。

76、CreateProcess与CreateThread

77、CloseHandle()，是减少线程的内核对象的引用计数，表明我们在主线程中对我们所创建的线程不感兴趣。当线程的内核对象引用计数为0时，就会清除线程。

78、

CreateMutex和ReleaseMutex(递减该线程对内核对象的引用计数)、配合WaitForSingleObject来实现线程同步

互斥对象

互斥对象(mutex)属于内核对象(这里说得内核对象应该是进程的内核对象，线程对其共享)，它能够确保线程拥有单个资源的互斥访问权。

互斥对象包含一个使用数量，一个线程ID和一个计数器。

ID用于标识系统中的哪个线程当前拥有互斥对象，计数器用于指明该线程拥有互斥对象的次数。

当某一线程终止时，就会把该线程对互斥对象的引用计数置为零，线程ID置为零。

把互斥对象理解为一把钥匙，线程理解为拥有这把钥匙的那个人（人的姓名相当于线程ID）。

79、可以用CreateMutex创建命名的互斥对象来实现只运行一个实例的功能。

80、AfxSocketInit()函数自动加载套接字库，进行版本的协商，是1.1版本的。Declared in Afxsock.h。

AfxSocketInit解释：

BOOL AfxSocketInit( WSADATA\* lpwsaData = NULL );

Return Value

Nonzero if the function is successful; otherwise 0.

Parameters

lpwsaData

A pointer to a [WSADATA](mk:@MSITStore:C:\Program%20Files\Microsoft%20Visual%20Studio\MSDN\2001OCT\1033\vcmfc.chm::/html/_mfc_wsadata_structure.htm) structure. If lpwsaData is not equal to NULL, then the address of the WSADATA structure is filled by the call to ::WSAStartup. This function also ensures that ::WSACleanup is called for you before the application terminates.

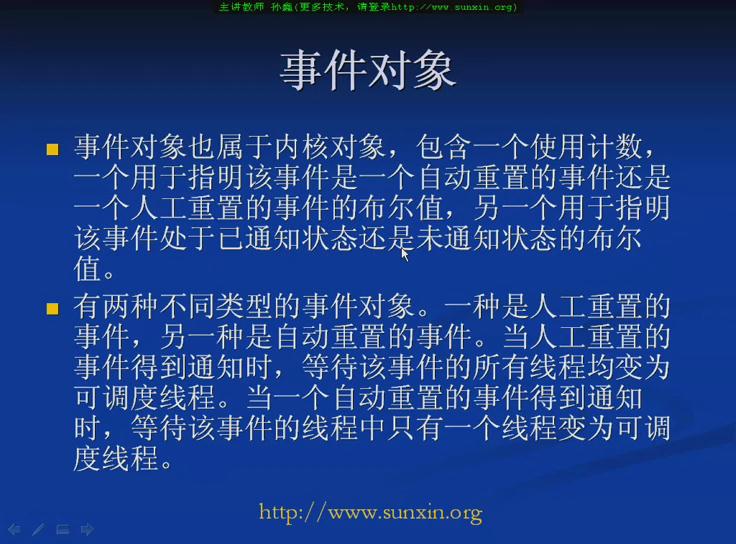
Remarks

Call this function in your CWinApp::InitInstance override to initialize Windows Sockets.

§Lesson16\_ThreadSynchronous

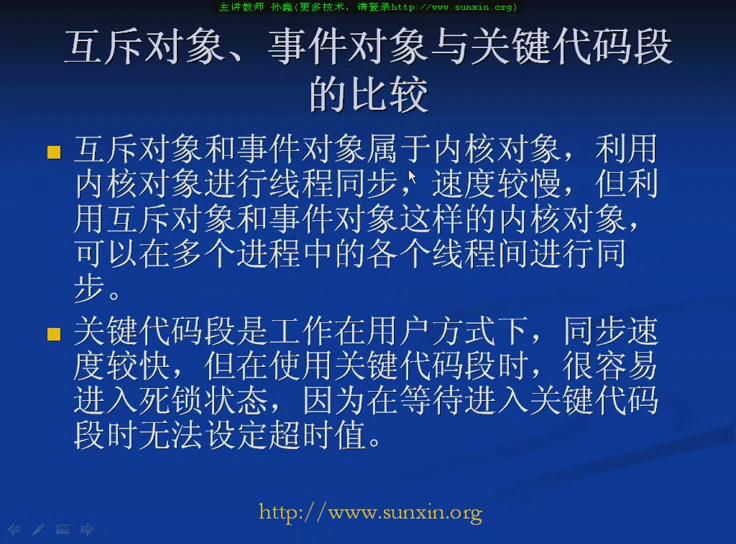
线程同步、异步套接字编程

81、CreateEvent ResetEvent SetEvent WatiForSingleObject



可以用CreateEvent创建命名的互斥对象来实现只运行一个实例的功能。

82、关键代码段（Critical Section），类似于现实生活中的公用电话厅。



我们在编写程序的时候，首选Critical Section，因为效率高，但是要注意死锁。要想更深入地了解多线程编程，请看《windows核心编程》。

83、

线程需要同步（阻塞），通信需要异步（非阻塞）（异步选择函数WSAAsyncSelect通过消息机制来做到）。

84、

扩展的socket函数：(需要手写2.2版本的winsock库)

WSAEnumProtocols

WSASocket

WSARecvFrom

WSASendTo

#include <winsock2.h>

…..

……….

WORD wVersionRequested;

WSADATA wsaData;

int err;

wVersionRequested = MAKEWORD( 2, 2 );

err = WSAStartup( wVersionRequested, &wsaData );

if ( err != 0 ) {

/\* Tell the user that we could not find a usable \*/

/\* WinSock DLL. \*/

return FALSE;

}

/\* Confirm that the WinSock DLL supports 2.2.\*/

/\* Note that if the DLL supports versions greater \*/

/\* than 2.2 in addition to 2.2, it will still return \*/

/\* 2.2 in wVersion since that is the version we \*/

/\* requested. \*/

if ( LOBYTE( wsaData.wVersion ) != 2 ||

HIBYTE( wsaData.wVersion ) != 2 ) {

/\* Tell the user that we could not find a usable \*/

/\* WinSock DLL. \*/

WSACleanup( );

return FALSE;

}

/\* The WinSock DLL is acceptable. Proceed. \*/

还需要链接一个库文件ws2\_32.lib

85、

|  |  |
| --- | --- |
| DWORD | 32-bit unsigned integer. |

指针间可以转换，比如char \* pch;然后使pch指向4字节的数据，可以把它转换成(DWORD)(DWORD\*)(pch)。

86、gethostbyname、gethostbyaddr

//gethostbyname 发送端通过主机名获取hostent结构体

//gethostbyaddr 接受端通过IP地址获取hostent结构体

§Lesson17\_ProcessCommunicate

进程间通信

86、

CreateFile函数不但能创建控制台（Console）、文件（File）、管道(Pipe)、油槽(Mailslot)，还能打开他们，所以功能很强大。

创建或者打开之后，就需要读写，可以调用ReadFile和WriteFile来完成。

最后调用CloseHandle来结尾。

87、

Clipboard:

给剪切板分配数据的时候，要多分配一个字节，因为把数据放到剪切板的时候，剪切板会自动把最后一个数据设置为’\0’，如果不多分配一个字节我们就会丢失一个数据。

88、

C语言中经常用得：malloc realloc memset memmove memcpy

Windows里面经常用得：GlobalAlloc FillMemory ZeroMemory MoveMemory CopyMemory

89、

GlobalLock可以把HANDLE -> POINTER，记得要用GlobalUnlock解锁。

90、

Pipe:

有匿名的和命名的

匿名的 CreatePipe,匿名管道的SECURITY\_ATTRIBUTES属性不能是NULL，因为匿名管道只能在父子进程之间通信，所以子进程要继承父进程的句柄。这个函数返回两个句柄，一个是读句柄，一个是写句柄。

命名的 CreateNamedPipe，这个函数返回一个管道句柄。

经典代码：

STARTUPINFO si;

ZeroMemory(&si,sizeof(STARTUPINFO));

si.cb=sizeof(STARTUPINFO);

si.dwFlags=STARTF\_USESTDHANDLES;

si.hStdInput=m\_hRead;

si.hStdOutput=m\_hWrite;

si.hStdError=GetStdHandle(STD\_ERROR\_HANDLE);

PROCESS\_INFORMATION pi;

if(!CreateProcess("..\\PipeChild\\Debug\\PipeChild.exe",NULL,NULL,NULL,TRUE,0,NULL,NULL,&si,&pi))

{

CloseHandle(m\_hRead);

CloseHandle(m\_hWrite);

m\_hRead=NULL;//

m\_hWrite=NULL;//主要是为了避免程序关闭时在析构函数里面再次关闭

MessageBox("创建子进程失败!");

return;

}

else

{

CloseHandle(pi.hProcess);//

CloseHandle(pi.hThread);//

//如果我们在父进程中不需要这两个内核对象，那么可以使

//父进程的计数器-1变为0

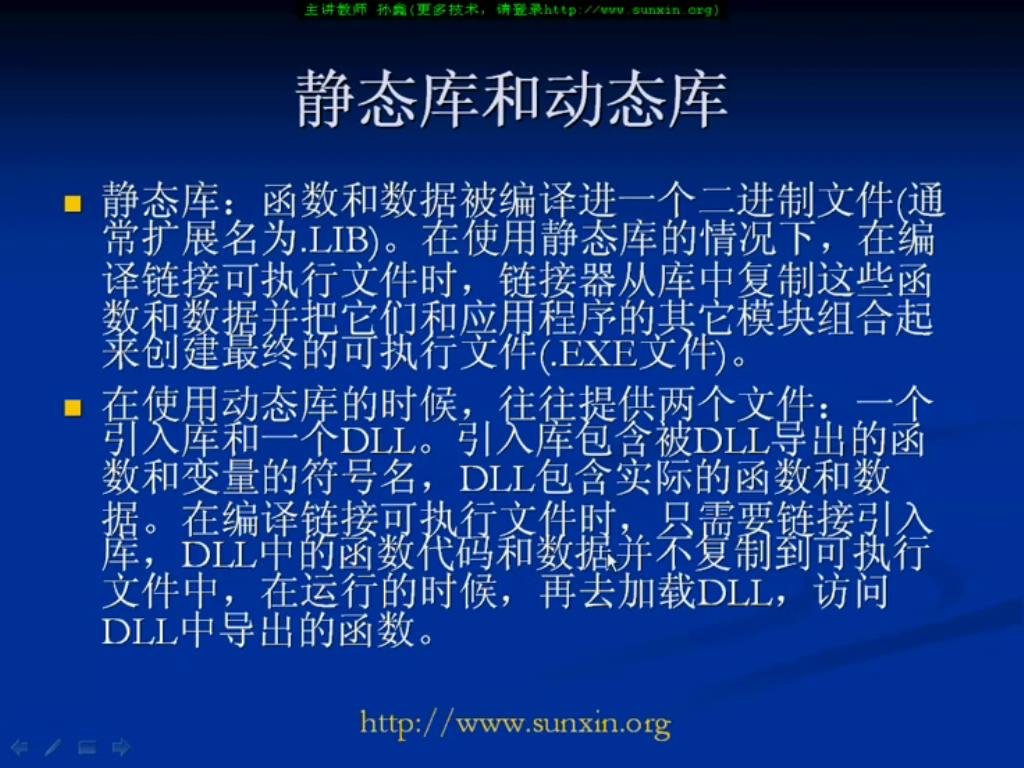
}

91、

Mailslot:CreateMailslot，mailslot在server端接收消息，即读取（ReadFile），在client端发送消息，即写入（WriteFile）。

§Lesson19\_DLL

92、静态库与动态库的区别



93、

dumpbin –exports Dll1.dll

dumpbin –imports DllTest.exe

94、

**GetForegroundWindow**

95、

LoadLibrary

DllMain