



第三章 网络安全编程基础

3

*内容提要



- Windows的内部机制
- C语言的四个发展阶段
- 实现Socket编程、注册表编程、定时器编程
- 驻留程序编程和多线程编程。

3.1 网络安全编程概述

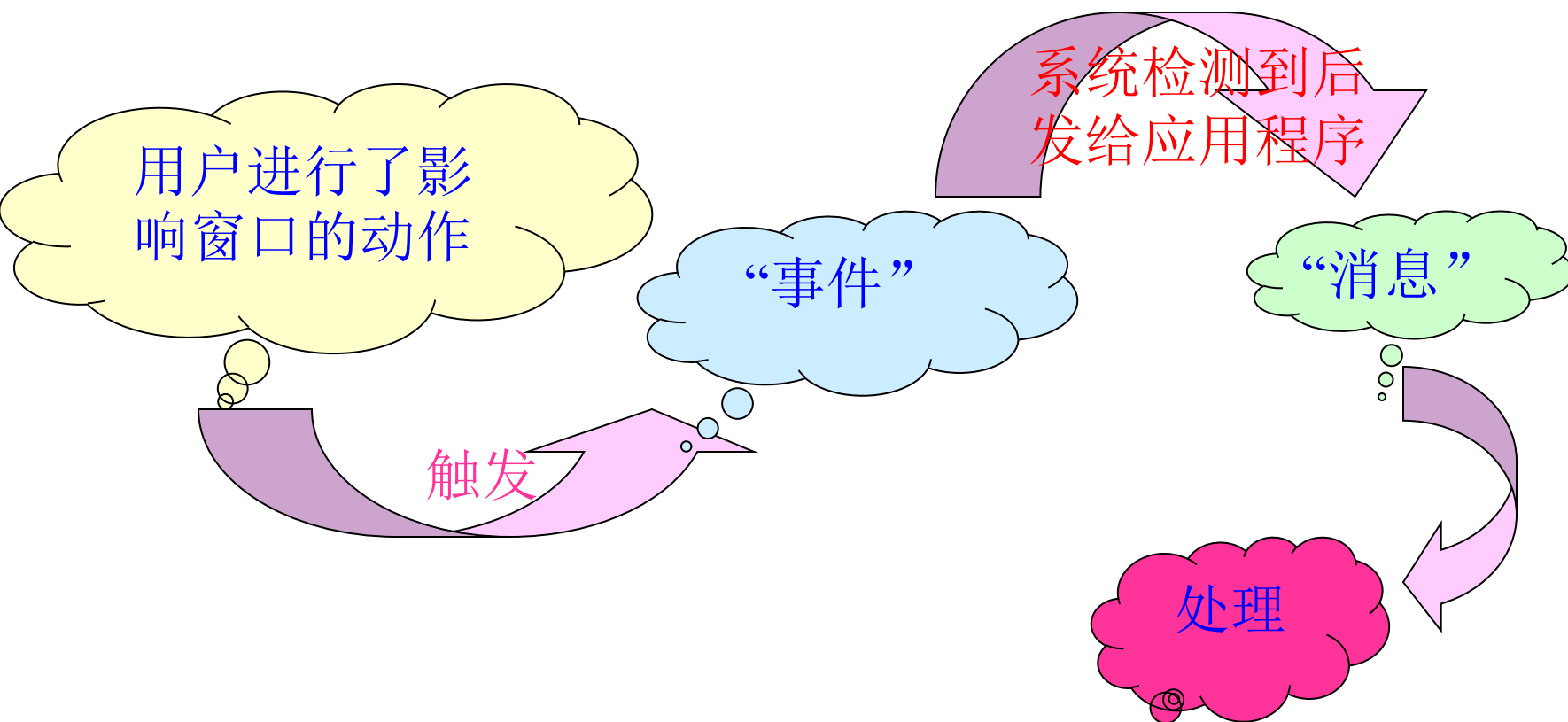


- C语言可以在Windows下编程，同样也可以在Linux下编程。
- 编程是一项比较综合的工作，除了熟练使用编程工具以外，还要了解系统本身的内部工作机理和编程语言。

3.1.1 Windows内部机制



- Windows是一个“基于事件的，消息驱动的”操作系统。



*八个基本概念



- 与Windows系统密切相关的八个基本概念分别是：
 - 窗口、程序、进程、线程
 - 消息、事件、句柄、API与SDK。

*窗口



- 窗口是Windows本身以及Windows 环境下的应用程序的基本界面单位

用户角度

- 窗口是显示在屏幕上的一个矩形区域
 - 具有标题栏、状态栏、最大化、最小化按钮的标准方框叫窗口，按钮也是特殊窗口
 - 是用户与生成该窗口的应用程序间的直观接口

- 窗口是受应用程序控制的一部分矩形屏幕区
 - 控制窗口的大小、风格、位置及内容

应用程序角度

*程 序



- 通常说的程序都是指一个能让计算机识别的文件
- 接触得最多的是以**exe**或者**com**作为扩展名的文件。
- 程序一组指令的集合
 - 例如:QQ程序, 主要包括
 - 界面外观控制指令集
 - **socket**网络通讯指令集
 - 数据信息存储指令集
- 可以任何语言形式表现: 高级语言、汇编语言、机器语言

*进 程



- 进程就是应用程序的执行实例（或称一个执行程序），进程是程序动态的描述。
 - 一个以exe作为扩展名的文件，在没有被执行的时候称之为应用程序，当用鼠标双击执行以后，就被操作系统作为一个进程执行了。
 - 当关机或者在任务栏的图标上单击鼠标右键选“退出”时，进程便消亡，彻底结束了生命。
- 进程经历了由“创建”到“消亡”的生命期，而程序自始至终存在于你的硬盘上，不管计算机是否启动。
- 查看当前进程如下图

*查看当前进程



Windows 任务管理器

文件(F) 选项(O) 查看(V) 帮助(H)

应用程序 进程 性能

映像名称	PID	CPU	CPU 时间	内存使用
LaunchAp. exe	1620	00	0:00:00	104 K
HotkeyApp. exe	1628	00	0:00:00	260 K
KeyHook. exe	1636	00	0:00:00	100 K
CtrlVol. exe	1652	00	0:00:00	104 K
Wbutton. exe	1656	00	0:00:00	116 K
RavMon. exe	1668	00	0:00:02	964 K
HTime. exe	1692	00	0:00:05	1,312 K
ctfmon. exe	1716	00	0:00:00	1,340 K
ymsgr_tray. exe	1800	00	0:00:00	100 K
kingamp2003. exe	1868	04	0:00:51	2,284 K
svchost. exe	1912	00	0:00:00	204 K
ravblmon. exe	2036	00	0:00:00	1,160 K
msdev. exe	2368	00	0:00:06	5,676 K
rvsim. exe	2424	00	0:00:00	952 K
editplus. exe	2828	00	0:00:10	5,952 K
notepad. exe	2860	00	0:00:00	1,144 K
taskmgr. exe	3592	00	0:00:00	4,076 K
HprSnap. exe	3616	04	0:00:01	3,260 K

结束进程 (E)

进程数: 57 CPU 使用: 14% 内存使用: 254144K / 630424K

*线程



- 线程是进程的一个执行单元，同一个进程中的各个线程对应于一组CPU指令、一组CPU寄存器以及一个堆栈。
- 可以从下载工具看出来
 - 原来的下载工具是单进程单线程
 - 目前的下载工具是单进程多线程

*消息



- 消息是应用程序和计算机交互的途径，在计算机上几乎做每一个动作都会产生一个消息
- 鼠标被移动会产生WM_MOUSEMOVE消息，
- 鼠标左键被按下会产生WM_LBUTTONDOWN的消息
- 鼠标右键按下便产生WM_RBUTTONDOWN消息

*事件、句柄



- 事件
 - 如在程序运行的过程中改变窗口的大小或者移动窗口等，都会触发相应的“事件”，从而调用相关的事件处理函数。
 - 例如：BUTTONCLICK事件，触发ONBUTTONCLICK()事件处理函数
- 句柄
 - 句柄是一个指针，通过句柄就可以控制该句柄指向的对象。
 - 编写程序总是要和各种句柄打交道的
 - 句柄是系统用来标识不同对象类型的工具，如窗口、菜单等，这些东西在系统中被视为不同类型的对象，用不同的句柄将他们区分开来。
 - 例如：HWND、HDC、HBRUSH、HMENU、HPEN等等

*API与SDK



- **API**是系统为应用程序提供的一系列函数接口。
 - 英文**Application Programming Interface** 的缩写，意思是“应用程序接口”，
- **SDK**是英文**Software Development Kit**的缩写，意思是“软件开发工具包”，
 - 微软提供了许多专门的**SDK**开发包，比如**DirectX**开发包和语音识别开发包等等。

3.1.2 学习Windows下编程

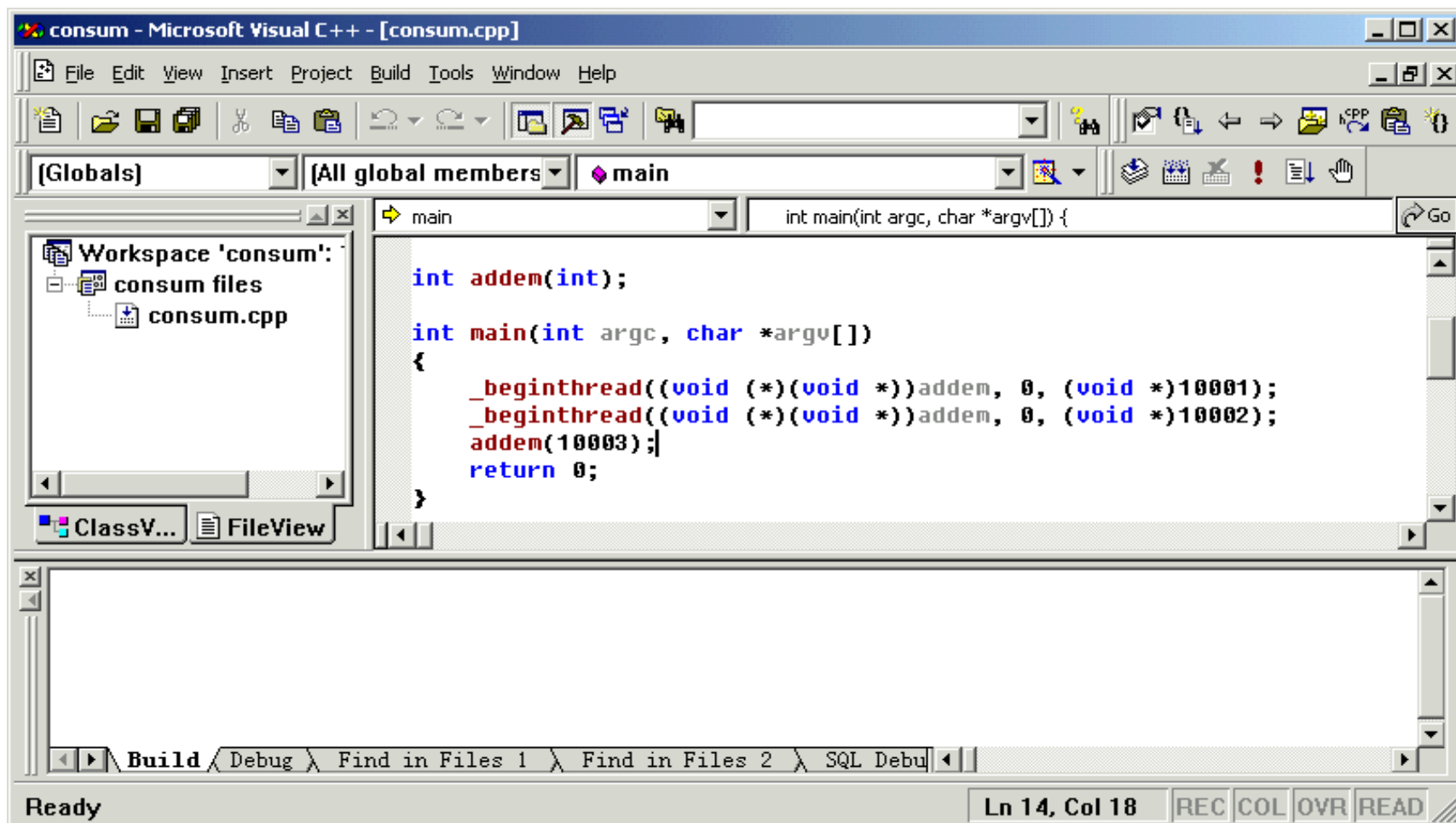


- 学习语言，选择语言和工具是第一步
 - 目前的编程语言有C、C++、C#、Java和汇编语言等等。
- 从实用的角度来讲，C/C++是最好的选择，微软公司的Visual C++是其相应开发工具

*开发工具



- 在开发工具上，选择比较流行的VC++6.0，而且最好是英文版本，主界面如图所示



*学习编程需要经历三大步



1、读程序

- 在没有阅读过一份完整的源代码之前，别指望能写出有多好的程序！

• 2、写程序

- “万丈高楼平底起”，编程贵在动手，只要动手去写就可以了。
- 还要依照自身的能力循序渐进地写

• 3、积累功能代码

- 将平时自己写的和自己已经读通的程序分类保存起来，建一个属于自己的代码库

3.1.3 选择编程工具



- 目前流行两大语法体系：**Basic**语系和**C**语系。同一个语系下语言的基本语法是一样。两大语系如图所示。

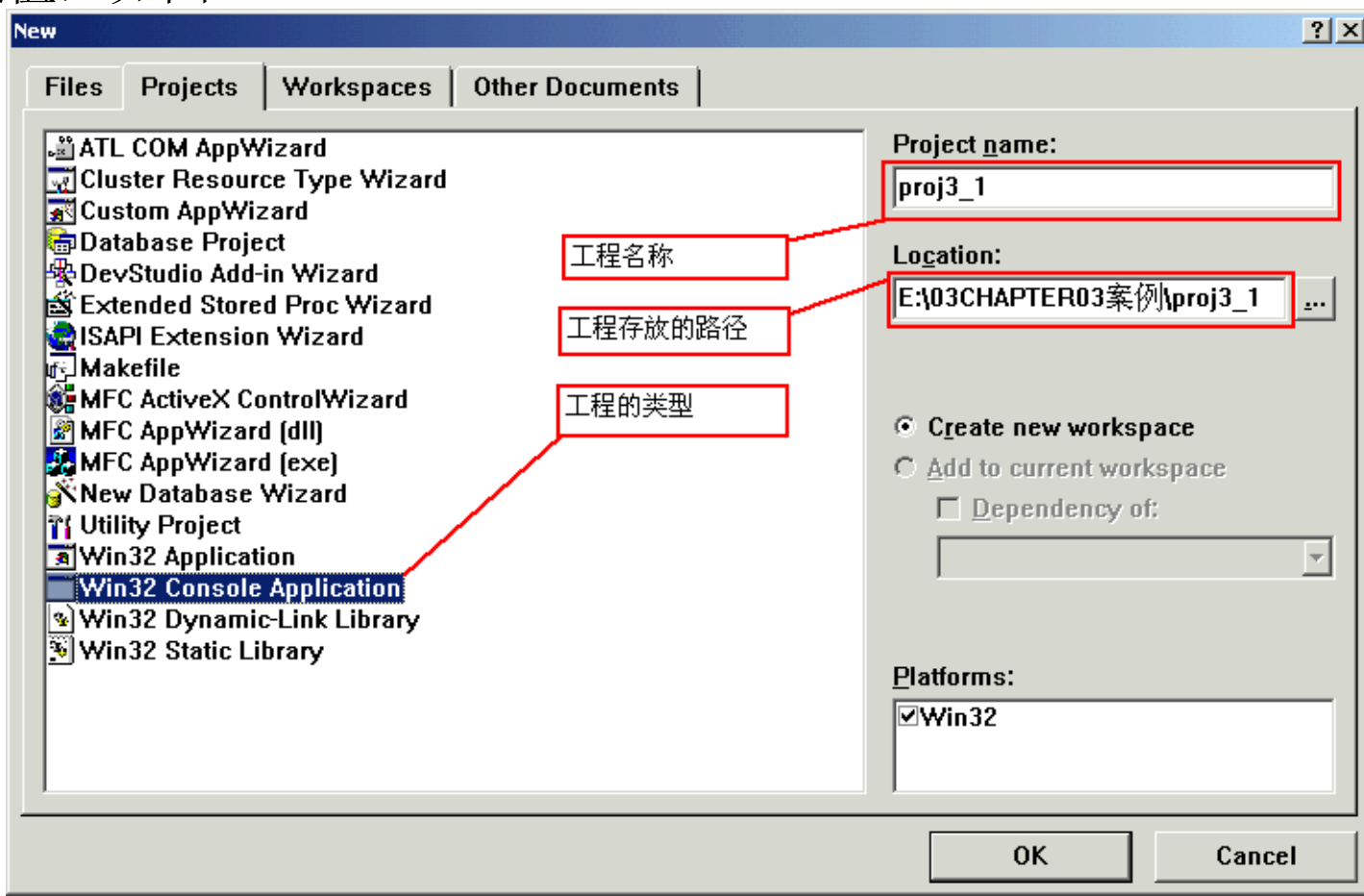
C语系：	C语言/C++语言/Java语言/Perl语言/C#/JavaScript
Basic语系：	Basic语言/VB6.0/VB.NET/VBScript/VBA

- **C**语系中，目前两大语言如日中天：**C++**和**Java**。
 - **C++**适宜做系统软件的开发、**Java**更适宜做网络应用开发。
 - 虽然**VC++.NET**已经面世很久了，但是**C++**的开发工具目前主流依然是**VC++6.0**和**C++ Builder6.0**。
 - **Java**流行的开发工具比较多，比如：**IBM**公司的**Visual Age**和**Websphere Studio**、**Eclipse**，**Insprise**公司**JBuilder**等等。

*VC++6.0



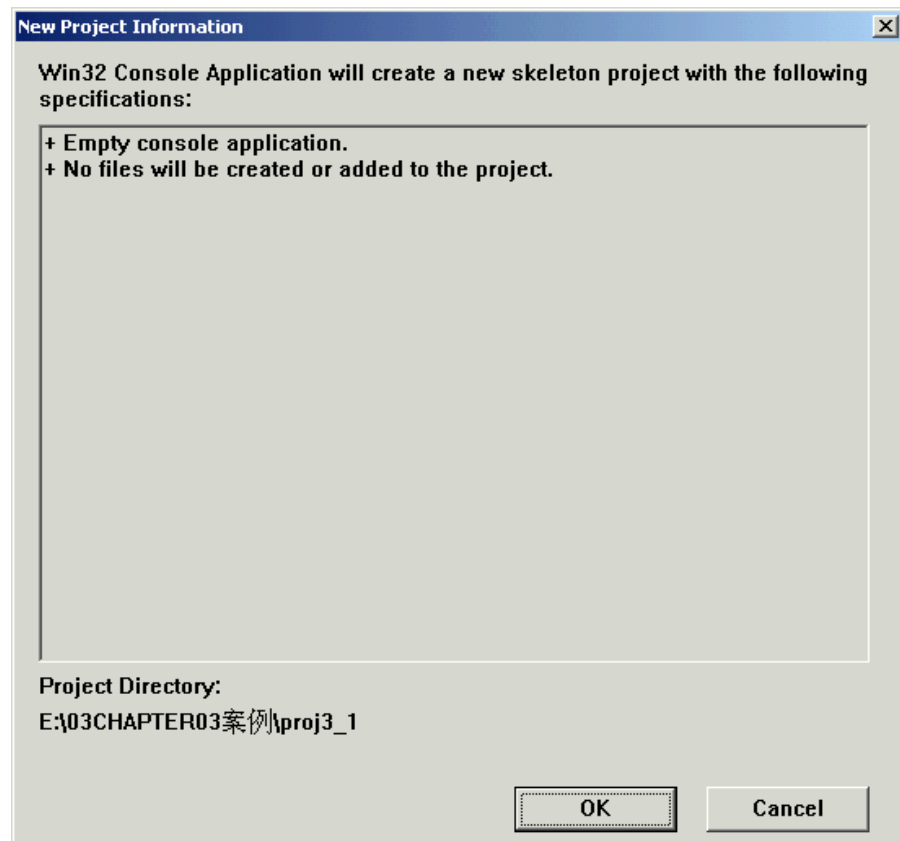
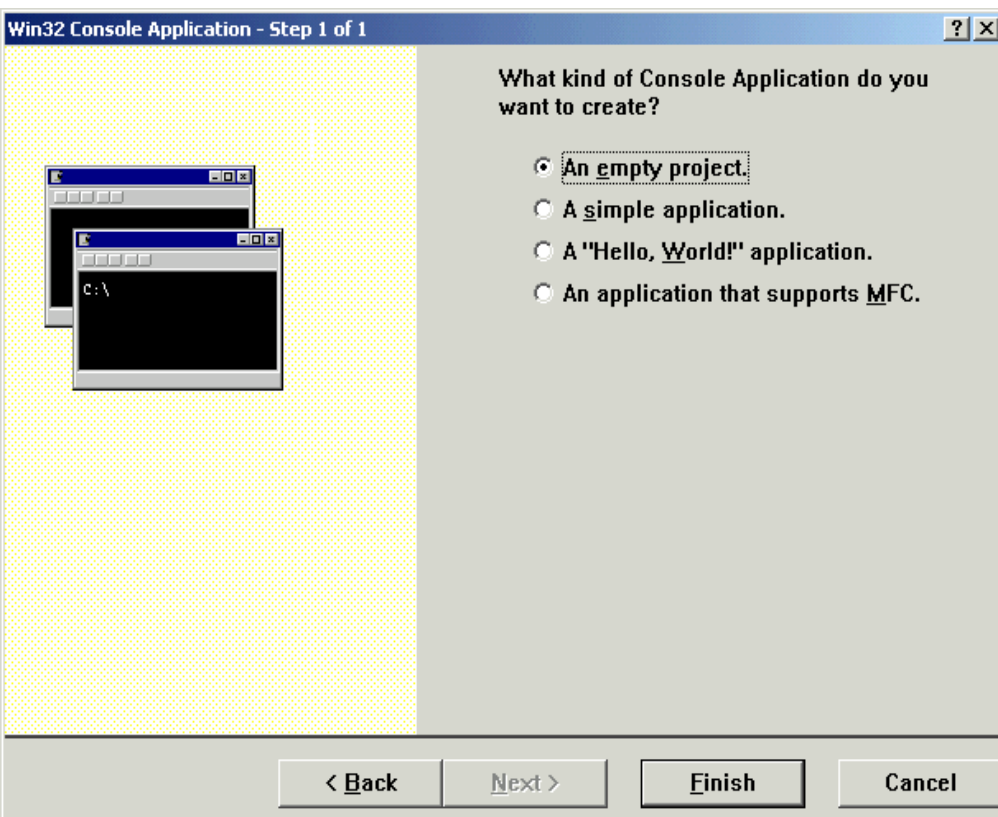
- 目前最常用的版本是VC++6.0。VC++有一套集成开发工具，其中包括各种编辑器、编译工具、集成调试器等等
- 下面通过一个程序来说明开发工具使用：File->New选择project，输入各项值，如图：



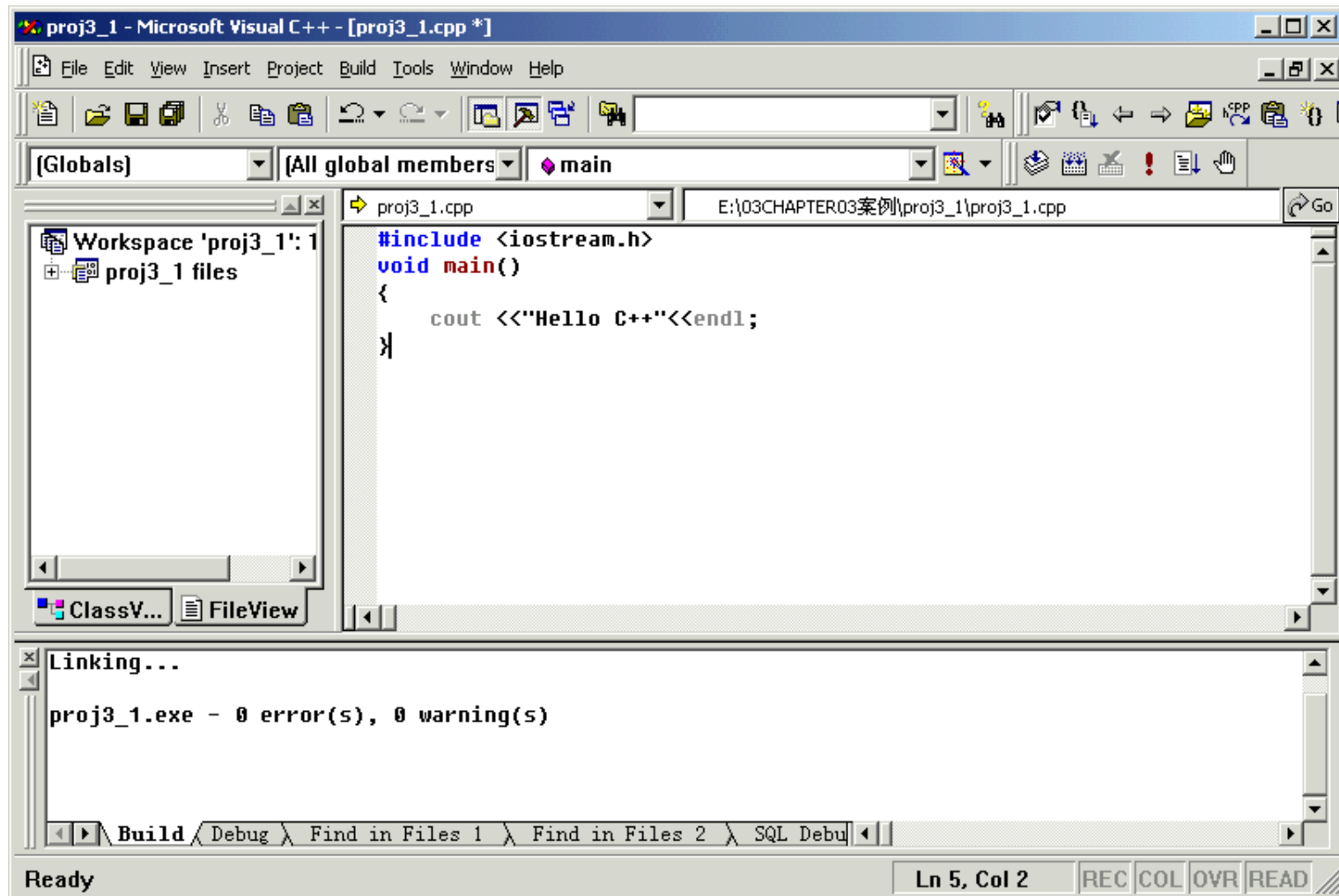
新建的是一个控制台程序



- 在左图的界面下选择创建工程的模板，选择空模板“An empty project”，点击按钮“Finish”，出现工程总结窗口，如右图所示。



- 为工程新加一个程序文件：File->New->FILES选择C++Source File，取名，后进入下面界面





- 选择 **BUILD->Execute**执行程序，出现结果

A screenshot of a Windows command prompt window. The title bar reads "E:\03CHAPTER03案例\proj3_1\Debug\proj3_1.exe". The window contains the text "Hello C++" followed by "Press any key to continue_" with a cursor. The window has standard Windows XP-style window controls (minimize, maximize, close) and a scrollbar on the right side.

```
"E:\03CHAPTER03案例\proj3_1\Debug\proj3_1.exe"  
Hello C++  
Press any key to continue_
```

*说明



- 程序proj3_1.cpp代码包括三行
- 第一行：“`#include <iostream.h>`”意思是引入C++的基本输入输出函数库，在C语言中引入的是“`stdio.h`”库。在`iostream.h`文件中定义了`cout`的功能是输出，`endl`的功能是回车换行。
- 第二行：“`void main()`”，`main()`函数是C/C++的主函数，`void`表示该函数没有返回值。
- 第四行：“`cout <<"Hello C++"<<endl;`”，“`cout<<`”功能是向屏幕输出。

3.2 C语言发展的四个阶段



- C语言经过不断的发展，在编程体系中可以将其分成四个阶段。
 - 1、面向过程的C语言。
 - 2、面向对象的C++语言。
 - 3、SDK编程。
 - 4、MFC编程（Microsoft Foundation Class：微软基类库）。

3.2.1 面向过程的C语言



- C语言功能非常强大
 - Linux/Unix操作系统就是用C语言写的，C语言直接调用操作系统提供的API函数可以编写非常强大的程序。
- C和C++的最主要区别是：C语言中没有类的概念，C++在C的语法基础上引入了类（Class）。
- 面向过程编程，最基本的程序用C语言编写如proj3_2.cpp所示。
- 案例名称：使用C语言编程
- 程序名称：proj3_2.cpp
- #include <stdio.h>
- main()
- {
- printf("Hello DOS\n");
- }

*案例3-1 读取命令行参数



- 案例名称：读取命令行参数

- 程序名称：proj3_3.cpp

参数个数

- `#include <stdio.h>`

- `int main(int argc, char *argv[.]).`

参数的值

- {

- `int i;`

- `for (i = 1; i < argc; i++)`

- {

- `printf("%s\n", argv[i]);`

- }

- `return 0;`

- }

编译后在命令行下运行如下



```
C:\WINNT\System32\cmd.exe
14 个目录    164,659,200 可用字节

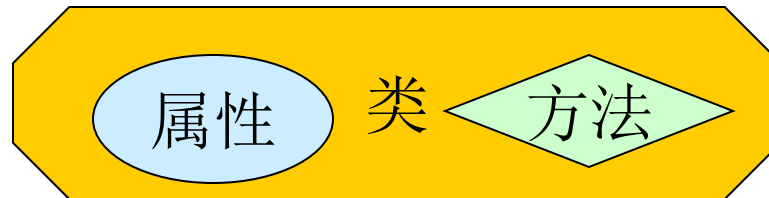
C:\>proj3_3 First Second Third
First
Second
Third

C:\>_
```

3.2.2 面向对象的C++语言



- 面向对象程序设计语言可以将一些变量和函数封装到**类（Class）**中
 - 当**变量**被类封装后，称之为**属性**或者数据成员
 - 当**函数**被类封装后，称之为**方法**或者**成员函数**



- 定义好的一个类，然后定义一个**类的实例**，这个实例就叫做**对象**，在C++中可以用类定义对象，使用方法如程序proj3_4.cpp所示。

*在C++中定义类



- 案例名称：在C++中使用类
- 程序名称：proj3_4.cpp

定义person类

```
#include <iostream.h>
class person
{
public:
    int heart;
    char *name;
    int run() //定义成员函数run()
    {
        heart=heart+20;
        return heart;
    }
};
```

private、protected

属性

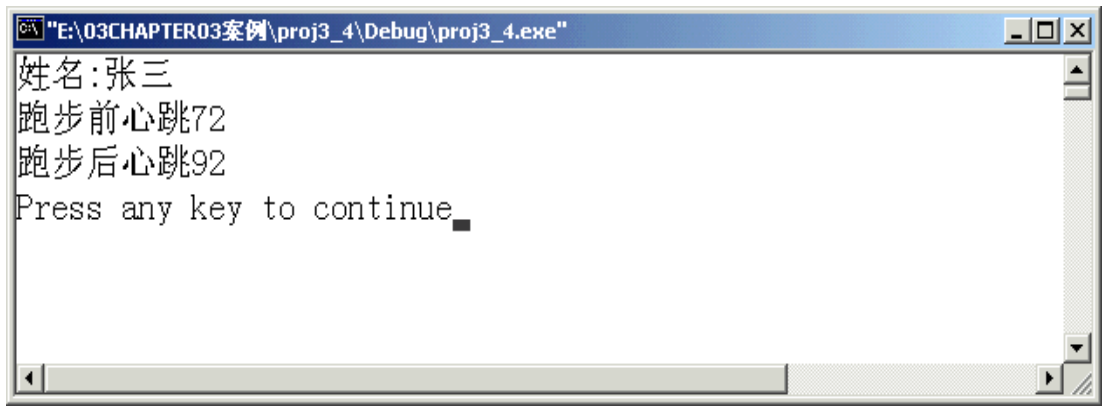
方法

*在C++中使用类



- void main()
- {
- int iRunStop;
- person ZhangSan;
- ZhangSan.name = "张三";
- ZhangSan.heart = 72;
-
- cout<<"姓名:"<< ZhangSan.name <<endl;
- cout<<"跑步前心跳"<< ZhangSan.heart <<endl;
- //run()为对象的方法
- iRunStop = ZhangSan.run();
- cout<<"跑步后心跳"<<iRunStop<<endl;
- }

- 编译运行输出结果如图



```
"E:\03CHAPTER03案例\proj3_4\Debug\proj3_4.exe"
姓名:张三
跑步前心跳72
跑步后心跳92
Press any key to continue
```

*将类定义与实现分离



- 类的定义放在.h文件中，实现放在.cpp文件中，具体如proj3_5所示
- 在proj3_5.h中有
- class person
- {
- public:
- int heart;
- char *name;
- int run() ;
- };

*将类定义与实现分离



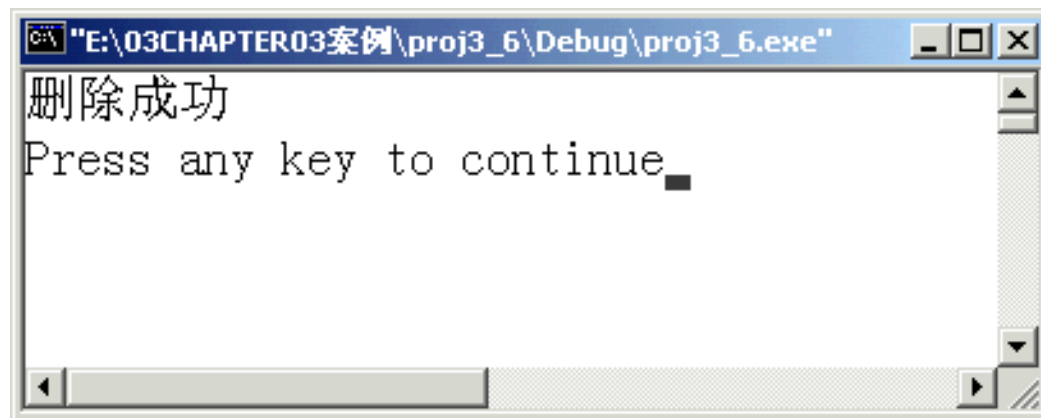
- 在proj3_5.cpp中有
- #include <iostream.h>
- #include "proj3_5.h"
- int person::run()
{
• heart=heart+20; return heart;
• }
- void main()
• { int iRunStop;
• person ZhangSan;
• ZhangSan.name = "张三"; ZhangSan.heart = 72;
• cout<<"姓名:"<< ZhangSan.name <<endl;
• cout<<"跑步前心跳"<< ZhangSan.heart <<endl;
• iRunStop = ZhangSan.run();
• cout<<"跑步后心跳"<<iRunStop<<endl;
• }

3.2.3 SDK编程



- C库提供了许多函数，可以直接拿来使用。比如利用C库提供的DeleteFile函数来删除一个文件，如程序proj3_6.cpp所示。
- 案例名称：调用C库函数
- 程序名称：proj3_6.cpp
-
- `#include <stdio.h>`
- `#include <windows.h>`
- `int main()`
- `{`
- `DeleteFile("C:\\test.txt");`
- `printf("删除成功\n");`
- `return 0;`
- `}`
- 执行结果如右图

C库函数提供



*编写窗口应用程序



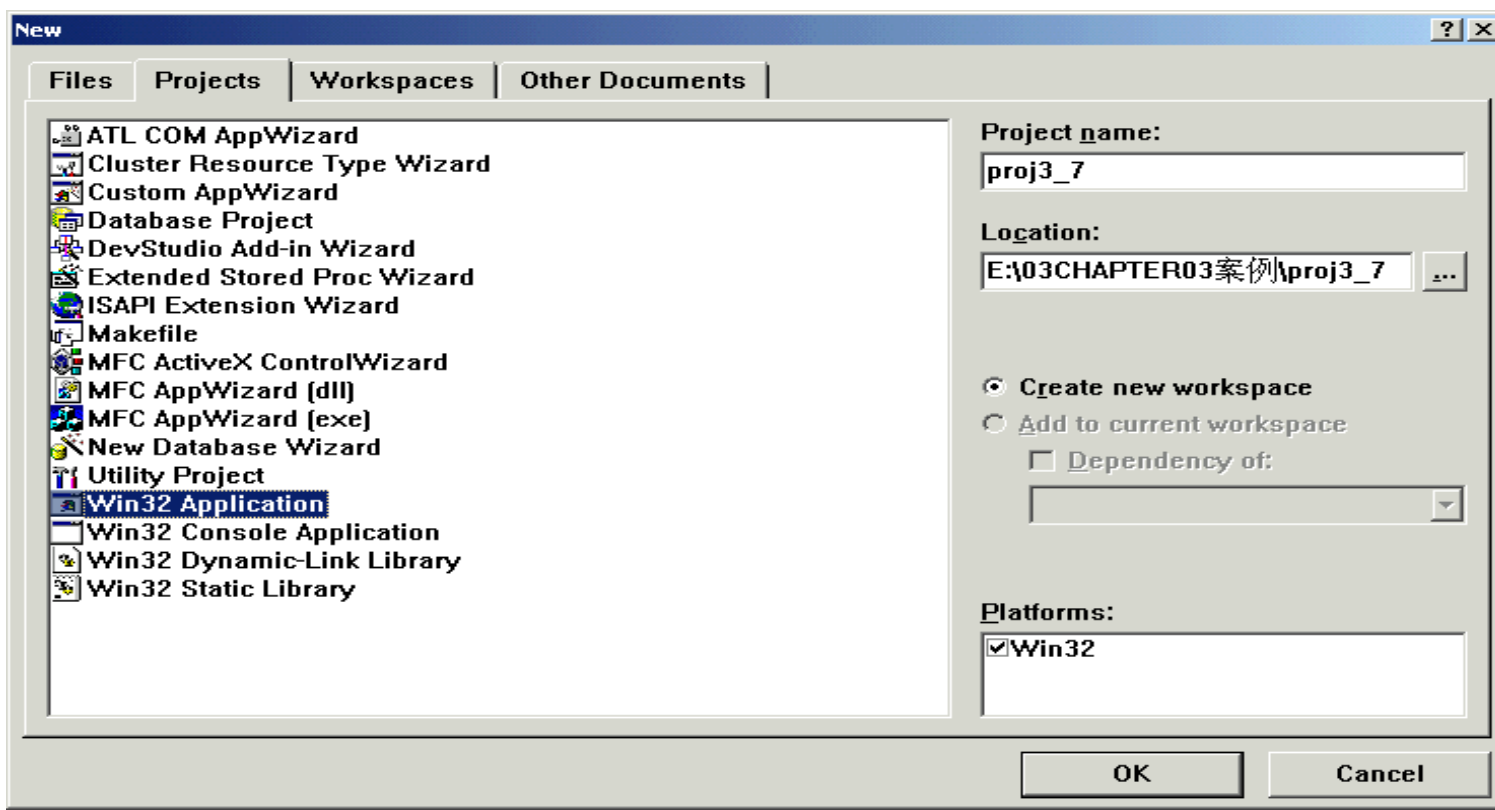
- 编写对话框的语法如下面程序
- 案例名称：编写对话框
- 程序名称：proj3_7.cpp
- `#include <windows.h>`
- `int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,`
- `PSTR szCmdLine, int iCmdShow)`
- `{`
- `MessageBox (NULL, TEXT ("Hello, Windows!"), TEXT ("HelloMsg"),`
- `MB_OK);`
- `return 0 ;`
- `}`
- 编译执行结果如图



*编写窗口应用程序



- 工程类型要改为WIN32 Application，如图
 - WIN32 Console Application——主函数为main()
 - WIN32 Application——主函数为WinMain()



*编写窗口应用程序



- **WinMain** (**HINSTANCE** **hInstance**, **HINSTANCE** **hPrevInstance**, **PSTR** **szCmdLine**, **int** **iCmdShow**)
- 参数一： **hInstance** 是当前实例的句柄。在这里**hInstance**表示应用程序本身。
- 参数二： **hPrevInstance**总是为**NULL**，在**Windows**早期版本中使用，在**32**位版本中，统一程序运行方式改变了，不再需要了！
- 参数三： **szCmdLine**是运行程序的命令行
- 参数四： **iCmdShow**用于指定程序窗口最初的显示模式，可以正常显示，也可以在初始化就最大化或者最小化。

案例3-2 利用SDK函数创建窗口



程序名称: proj3_8.cpp

```
#include <windows.h>
```

```
WNDCLASS wc;
```

```
HWND h_wnd;
```

```
MSG msg;
```

窗口对象

窗口句柄

Windows消息对象

```
/* 消息处理函数wndProc的声明*/
```

```
long WINAPI WindowProc(HWND,UINT,WPARAM,LPARAM);
```

案例3-2 利用SDK函数创建窗口



- /* winMain 函数的声明*/
- int PASCAL WinMain(HINSTANCE h_CurInstance,HINSTANCE h_PrevInstance,LPSTR p_CmdLine,int m_Show)
- {
- /*初始化wndclass结构变量*/
- wc.lpfWndProc = WindowProc;
- wc.hInstance = h_CurInstance;
- wc.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
- wc.lpszClassName = "TheMainClass";
- /* 注册WndClass结构变量*/
- RegisterClass(&wc);

案例3-2 利用SDK函数创建窗口



- /* 创建窗口*/

- h_wnd=CreateWindow(
• "TheMainClass",
• "Our first Window",
• WS_OVERLAPPEDWINDOW
• 0,0,
• 400,500,
• 0,0,
• h_CurInstance,
• 0);

注册的窗口

窗口标题

窗口样式

窗口左上角的位置

窗口宽度与高度

程序当前句柄

- /* 显示窗口*/

- ShowWindow(h_wnd,SW_SHOWMAXIMIZED);

- /*消息循环*/

- while(GetMessage(&msg,NULL,0,0))

- DispatchMessage(&msg);

- return (msg.wParam);

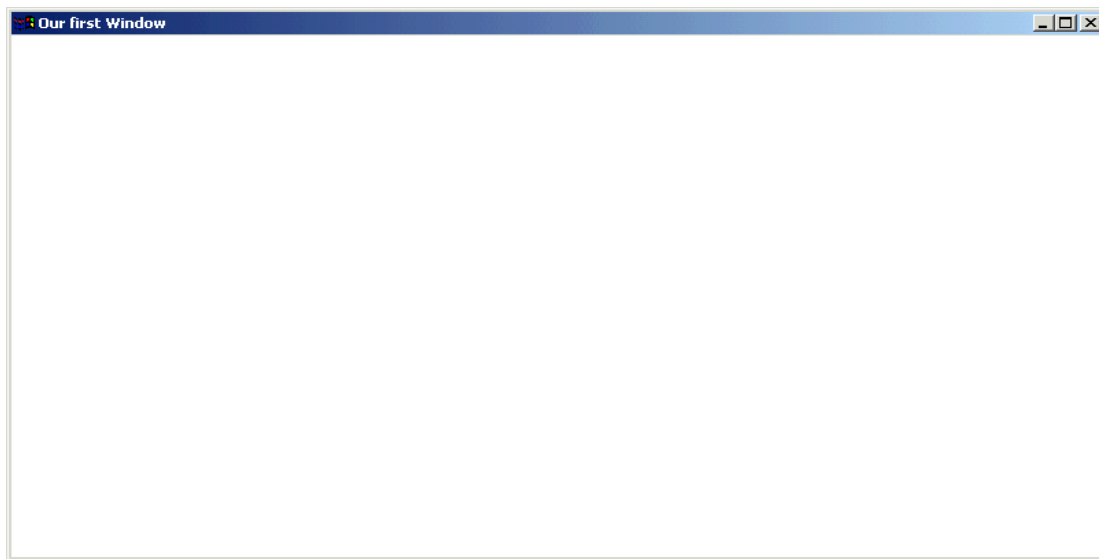
- }

将消息发送给
Windowproc

案例3-2 利用SDK函数创建窗口



- /* 定义消息处理函数*/
- long WINAPI WindowProc(HWND h_wnd,UINT WinMsg,WPARAM w_param,LPARAM l_param)
- {
- if(WinMsg==WM_DESTROY)
- PostQuitMessage(0);
- return
- DefWindowProc(h_wnd,WinMsg,w_param,l_param);
- }
- 执行结果如右图



3.2.4 MFC编程



- SDK的功能非常强大，需要记很多的函数
- 微软将SDK的函数分类进行封装，这样就诞生了MFC（Microsoft Foundation Class）。
- MFC程序的最基本的程序骨架如proj3_9.cpp所示。

3.2.4 MFC编程



类继承

- `#include<afxwin.h>`
- `class sample:public CFrameWnd` ○ ○ ○
- `{`
- `public:`
- `sample() //构造函数`
- `{`
- `Create(NULL,"My Window");`
- `MessageBox("My Window","CFrame constructor");`
- `}`
- `};`
-
- `class App:public CWinApp`
- `{`
- `public:`
- `BOOL InitInstance();`
- `BOOL ExitInstance();`
- `};`
-

3.2.4 MFC编程

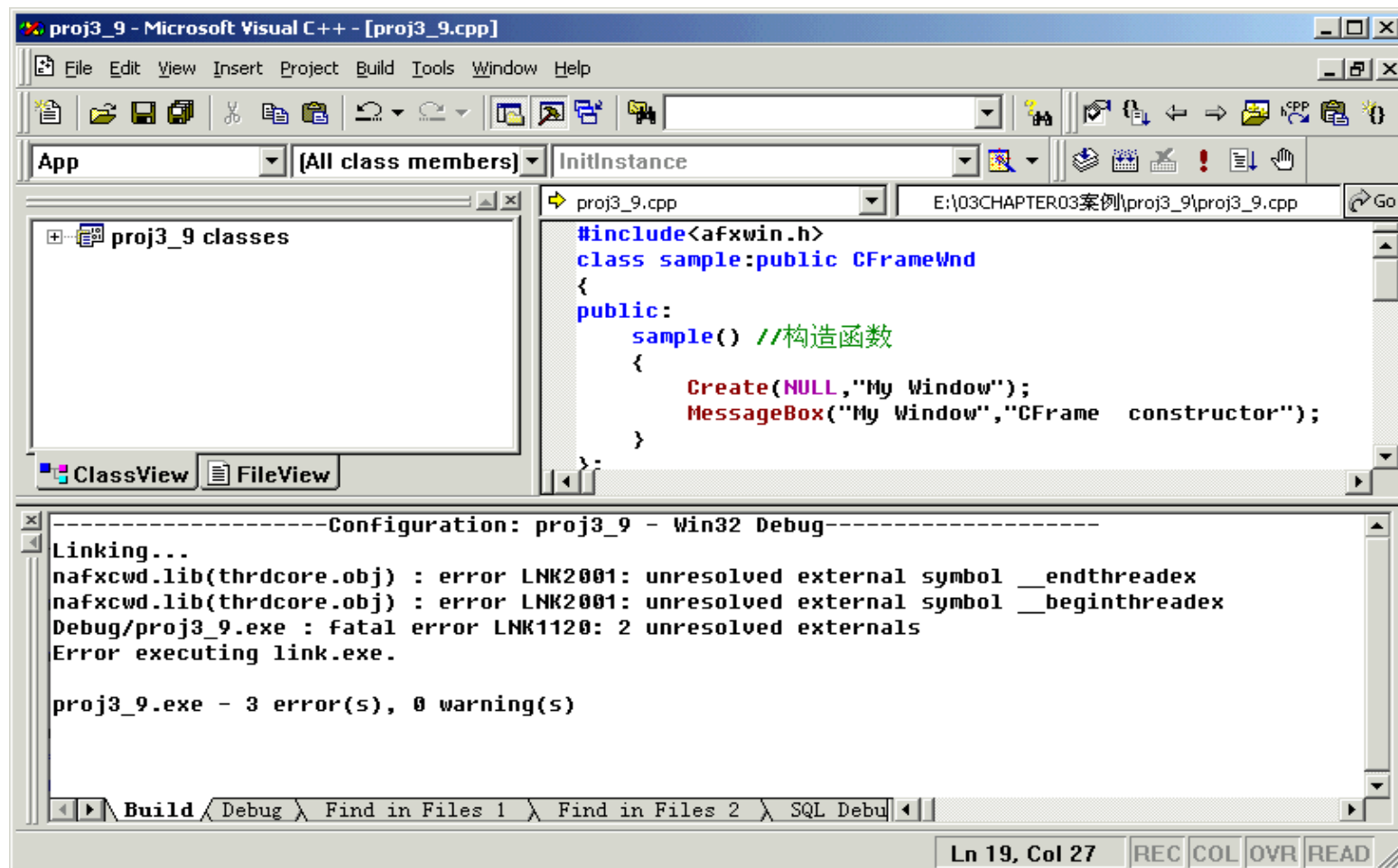


- `//InitInstance函数的定义//`
- `BOOL App::InitInstance()`
- `{`
- `MessageBox(0,"My Window","InitInstance",MB_OK|MB_ICONASTERISK);`
-
- `sample *obj;`
- `obj=new sample;`
- `m_pMainWnd=obj;`
- `obj->ShowWindow(SW_SHOWMAXIMIZED);`
-
- `return TRUE;`
- `}`
- `//ExitInstance函数定义`
- `BOOL App::ExitInstance()`
- `{`
- `MessageBox(0,"My Window","ExitInstance", MB_OK|MB_ICONHAND);`
- `return TRUE;`
- `}`
- `//创建应用程序对象`
- `App appobject;`

3.2.4 MFC编程



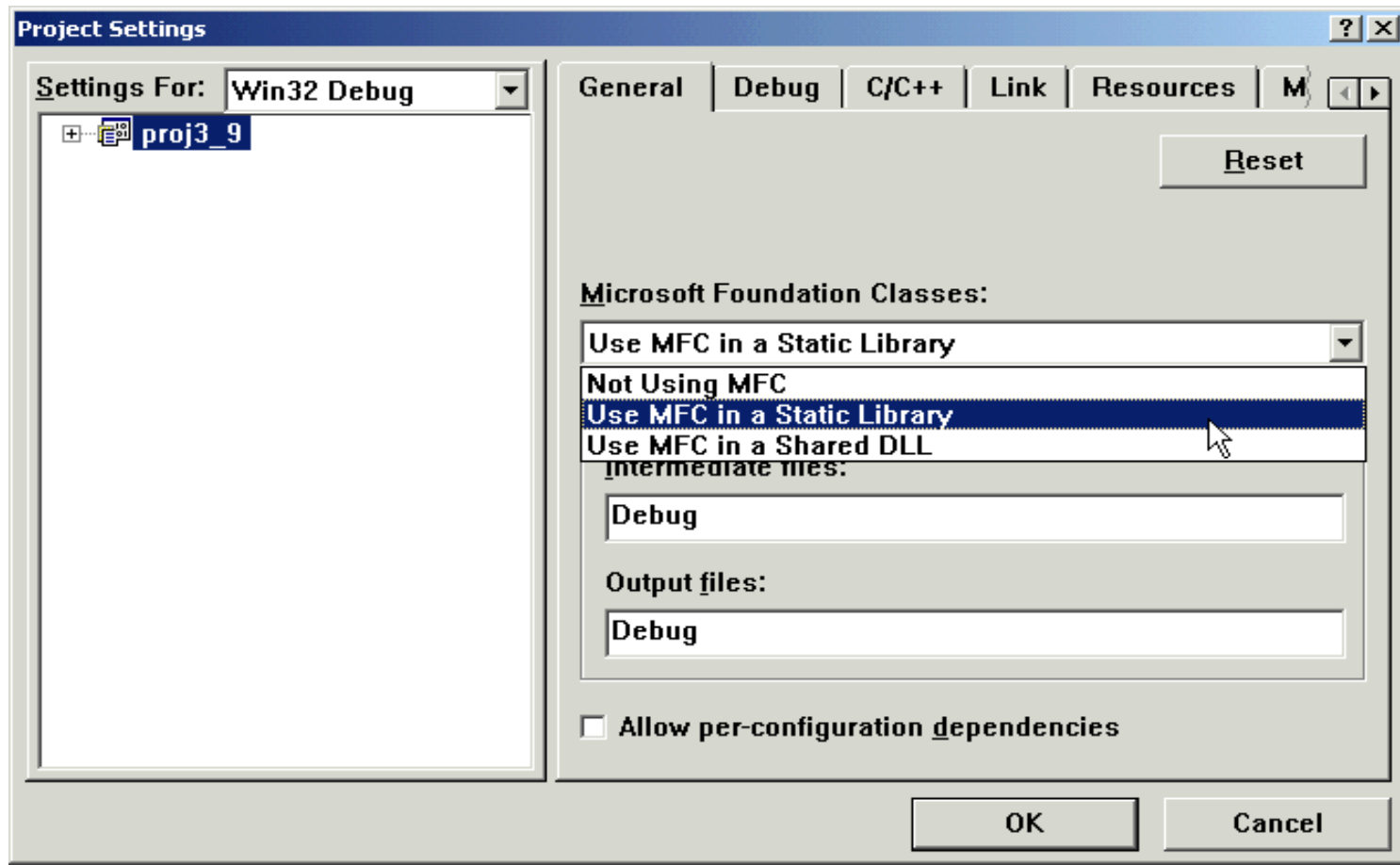
- 如果配置的执行方式与前面一样，则会出错



3.2.4 MFC编程



- 两种解决办法
 - 新建工程时选择 MFC AppWizard(exe)
 - 在原来基础上，project->setting设置中修改，如图



3.2.4 MFC编程



- 程序首先执行最一行: `App appobject;`
- 申明了一个App对象实例,将自动调用`InitInstance()`
- 执行第一句

```
MessageBox(0,  
    "My Window",  
    "InitInstance",  
    MB_OK|MB_ICONASTERISK);
```

弹出第一个对话框



3.2.4 MFC编程



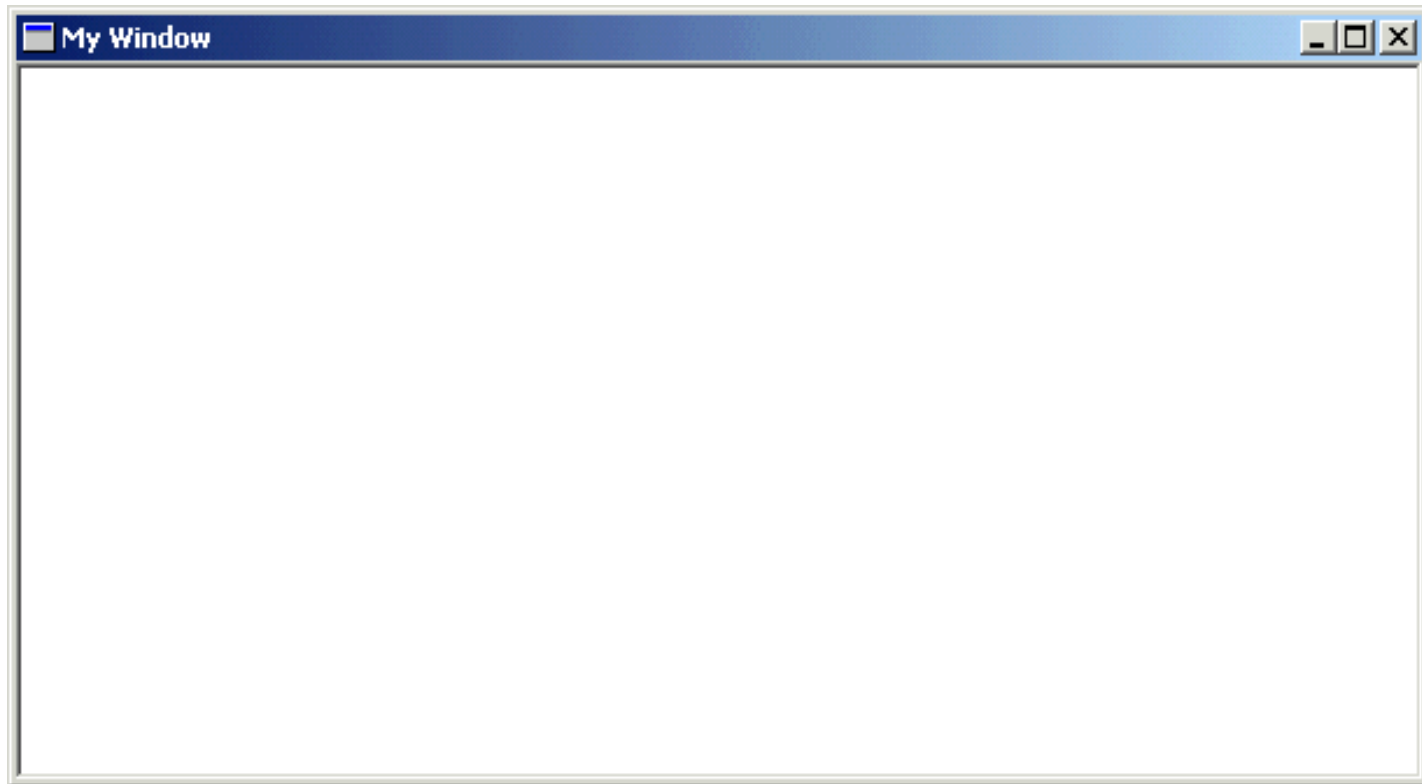
- 接着程序执行 `sample *obj; obj=new sample;`
- 当Sample类对象实例化时自动调用构造函数
- `sample() //构造函数`
- `{`
- `Create(NULL,"My Window");`
- `MessageBox("My Window","CFrame`
- `constructor");`
- `}`
- 弹出对话框



3.2.4 MFC编程



- 接着程序执行 `m_pMainWnd=obj;`
- `obj->ShowWindow(SW_SHOWMAXIMIZED);`
- 显示主窗口(最大化)



3.2.4 MFC编程



- 当退出主窗口时,系统自动调用ExitInstance()
- 执行MessageBox(0,
- "My Window",
- "ExitInstance",
- MB_OK|MB_ICONHAND);
-
- 弹出窗口如右图



案例3-3 MFC的事件处理机制



```
• #include<afxwin.h>
• class sample:public CFrameWnd
• {
• public:
•     sample() //构造函数
•     {
•         Create(NULL,"My Window");
•     }
•     void OnLButtonDown(UINT,CPoint)
•     {
•         MessageBox("You clicked the left Mouse Button","Hello world",0);
•     }
•     void OnRButtonDown(UINT,CPoint)
•     {
•         MessageBox("You clicked the right Mouse Button","Hello world",0);
•     }
•     DECLARE_MESSAGE_MAP()
• };
•
```

案例3-3 MFC的事件处理机制



- **BEGIN_MESSAGE_MAP**(sample,CFrameWnd)
- ON_WM_LBUTTONDOWN()
- ON_WM_RBUTTONDOWN()
- **END_MESSAGE_MAP**()
- **class** App:**public** CWinApp
- {
- **public:**
- BOOL InitInstance();
- BOOL ExitInstance();
- };
- **//InitInstance函数的定义//**
- **BOOL** App ::InitInstance()
- {
- sample *obj;
- obj=new sample;
- m_pMainWnd=obj;
- obj->ShowWindow(SW_SHOWMAXIMIZED);
- return TRUE;
- }

案例3-3 MFC的事件处理机制



- `//ExitInstance`函数定义
- `BOOL App::ExitInstance()`
- `{`
- `return TRUE;`
- `}`
- `//创建应用程序对象`
- `App appobject;`
- 编译运行会在按下左键，右键时弹出对话框

3.3 网络安全编程



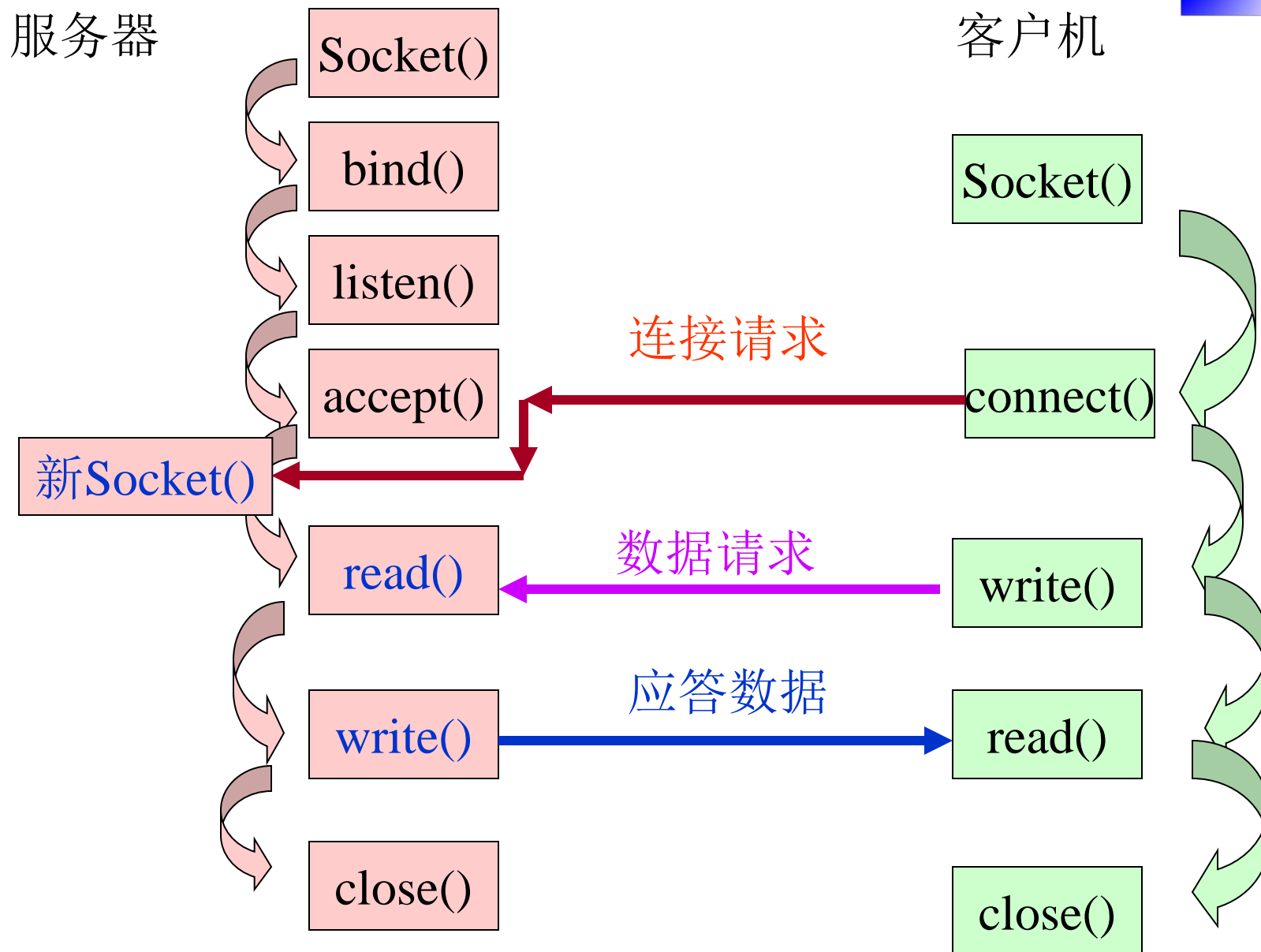
- 网络安全基础编程技术主要包括6个方面：
 - Socket编程
 - 注册表编程
 - 文件系统编程
 - 定时器编程
 - 驻留程序编程
 - 多线程编程。

3.3.1 Socket编程



- 谈网络安全编程离开网络编程就会大失其味，凡是基于网络应用的程序都离不开Socket。
- Socket的意思是套接字，是计算机与计算机之间通信的接口。
- Socket网络编程一般采用服务器/客户机模式，有两种不同的套接字
 - 流套接字
 - 数据报套接字

流套接字的编程时序图如下

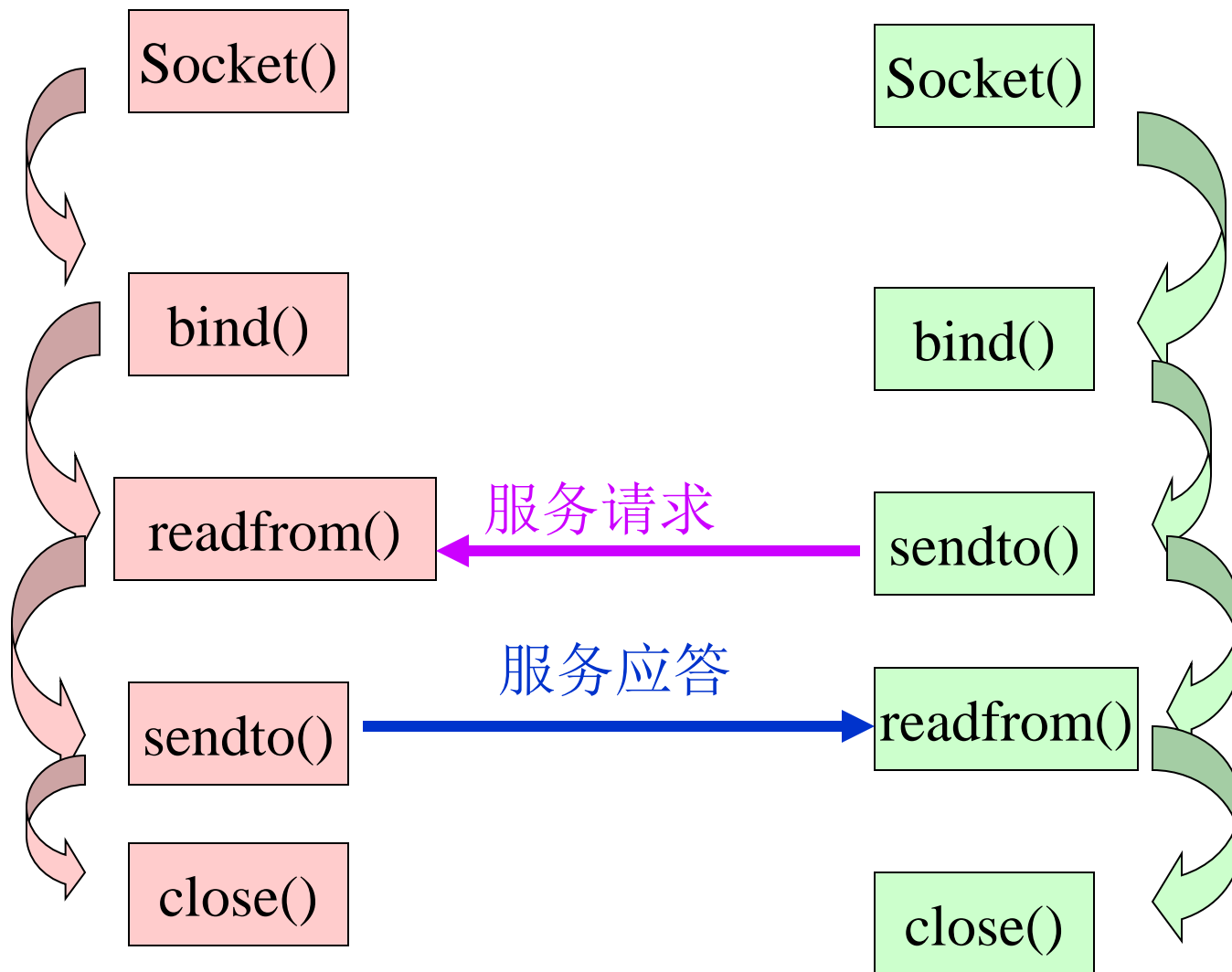


数据报套接字的编程时序图如下



服务器

客户机



3.3.1 Socket编程



- 使用Winsock提供的API函数是最基本的网络编程技术
- 程序proj3_11.cpp利用Socket获得本机的IP地址和机器名
- 附加：利用socket编程实现扫描网站端口

3.3.1 Socket编程



- 案例名称：使用Socket得到IP地址
- 程序名称：proj3_11.cpp
-
- `#include <winsock.h>`
- `#include <stdio.h>`
- `void CheckIP(void)` //CheckIP函数，用于获取本机IP地址
- {
- `WORD wVersionRequested;`//用于存放Winsock版本的值
- `WSADATA wsaData;`
-
- `char name[255];`//用于存放主机名
- `PHOSTENT hostinfo;`
- `wVersionRequested = MAKEWORD(2, 0);`
- //调用MAKEWORD()函数获得Winsock的版本，用于加载Winsock
- 库
-

3.3.1 Socket编程



```
• if ( WSAStartup( wVersionRequested, &wsaData ) == 0 )
• {
•     //加载Winsock库，如果WSAStartup()函数的返回值为0，说明加载成功
•     if( gethostname ( name, sizeof(name)) == 0)
•     {
•         //判断是否成功的将本地主机名存放入由name参数指定的缓冲区中
•         if((hostinfo = gethostbyname(name)) != NULL)
•         {
•             //如果获得主机名成功的话，调用inet_ntoa()函数取得IP地址
•             LPCSTR ip = inet_ntoa (
•                 *(struct in_addr *)*hostinfo->h_addr_list);
•             printf("本机的IP地址是: %s\n",ip); //输出IP地址
•             printf("本机的名称是: %s\n",name);
•         }
•     }
•     WSACleanup( ); //卸载Winsock库，并释放所有资源
• }
•
• int main()
• {
•     CheckIP(); //调用CheckIP()函数获得并输出IP地址
•     return 0;
• }
```

3.3.1 Socket编程



由于采用main()函数，所以工程采用“Win32 Console Application”

- 编译执行，出现错误，如图

The screenshot shows the Microsoft Visual C++ 6.0 IDE. The main window displays the source code for 'proj3_11.cpp'. The code includes headers for winsock.h, stdio.h, stdlib.h, and string.h. It defines a 'CheckIP' function and declares a 'WORD' variable 'wVersionRequested' and a 'WSADATA' variable 'wsaData'. It also declares a 'char' array 'name' and a 'PHOSTENT' variable 'hostinfo'. The bottom window shows the output of the linker, indicating several unresolved external symbols: '_WSACleanup@0', '_inet_ntoa@4', '_gethostbyname@4', '_gethostname@8', and '_WSAStartup@8'. The linker reports a fatal error LNK1120: 5 unresolved externals. The status bar at the bottom indicates 'Ln 13, Col 40'.

```
proj3_11 - Microsoft Visual C++ - [proj3_11.cpp]
File Edit View Insert Project Build Tools Window Help
[Globals] [All global members] main
proj3_11 classes
ClassView FileView
proj3_11.cpp E:\03CHAPTER03案例\proj3_11\proj3_11.cpp
#include <winsock.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void CheckIP(void) //CheckIP函数，用于获取本机IP地址
{
    WORD wVersionRequested; //WORD类型变量，用于存放Winsock库
    WSADATA wsaData;

    char name[255]; //用于存放主机名
    PHOSTENT hostinfo;
}

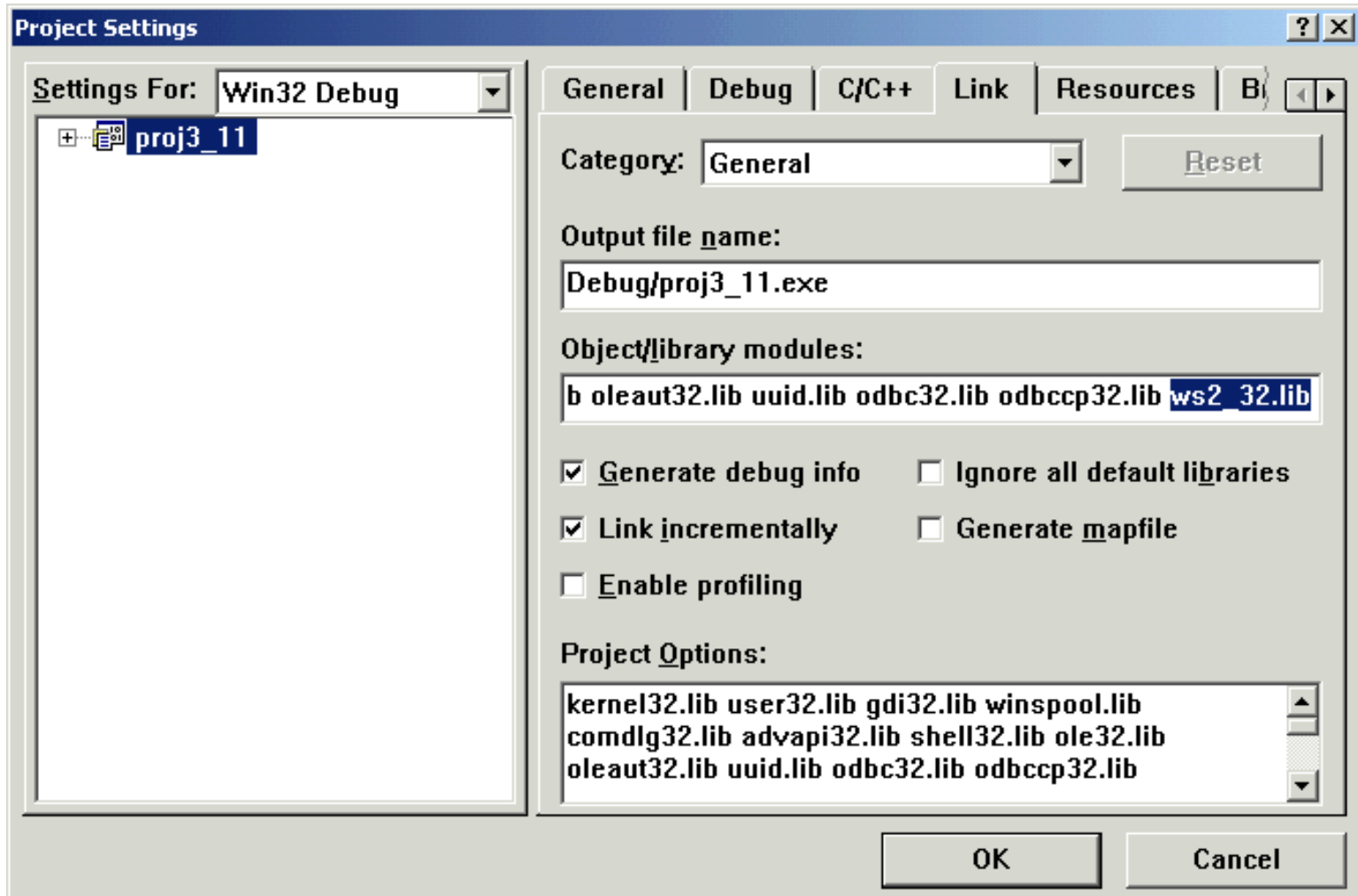
proj3_11.cpp
Linking...
proj3_11.obj : error LNK2001: unresolved external symbol _WSACleanup@0
proj3_11.obj : error LNK2001: unresolved external symbol _inet_ntoa@4
proj3_11.obj : error LNK2001: unresolved external symbol _gethostbyname@4
proj3_11.obj : error LNK2001: unresolved external symbol _gethostname@8
proj3_11.obj : error LNK2001: unresolved external symbol _WSAStartup@8
Debug/proj3_11.exe : fatal error LNK1120: 5 unresolved externals
Error executing link.exe.

proj3_11.exe - 6 error(s), 0 warning(s)
Build Debug Find in Files 1 Find in Files 2 SQL Debug
Ready Ln 13, Col 40 REC COL OVR READ
```

3.3.1 Socket编程



- 消除错误的方法是在project->settings->link->Object/library modules,加入“WS_32.lib”



3.3.1 Socket编程



- 再编译执行就可得到

```
"E:\03CHAPTER03案例\proj3_11\Debug\proj3_11.exe"  
本机的IP地址是: 172.18.25.110  
本机的名称是: szg  
Press any key to continue
```

*利用Socket实现简单扫描器



- 程序清单如下：
- `#include <winsock2.h>`
- `#include "stdio.h"`
- `#pragma comment(lib,"ws2_32")`
- `#include <stdlib.h>`
- `#include <windows.h>`
- `void main()`
- `{`
 - `WSADATA ws;`
 - `SOCKET s;`
 - `struct sockaddr_in addr;`
 - `int RESULT;`
 - `long lRESULT;`

*利用Socket实现简单扫描器



```
for (int i=1;i<200;i++)
{
    IRESULT=WSAStartup(0x0101,&ws);
    S=socket(PF_INET,SOCK_STREAM,0);
    addr.sin_family=PF_INET;
    addr.sin_addr.s_addr=inet_addr("162.105.195.158");
    addr.sin_port=htons(i);
    if (S==INVALID_SOCKET) break;
    RESULT=connect(S,(struct sockaddr*)&addr,sizeof(addr));
    if(RESULT!=0)//连接失败，表明该端口没开放
    { printf("162.105.195.158:%i  inactive\n",i); WSACleanup(); }
    else
    { printf("162.105.195.158:%i  active\n",i);      }
    closesocket(s);
}
• }
```

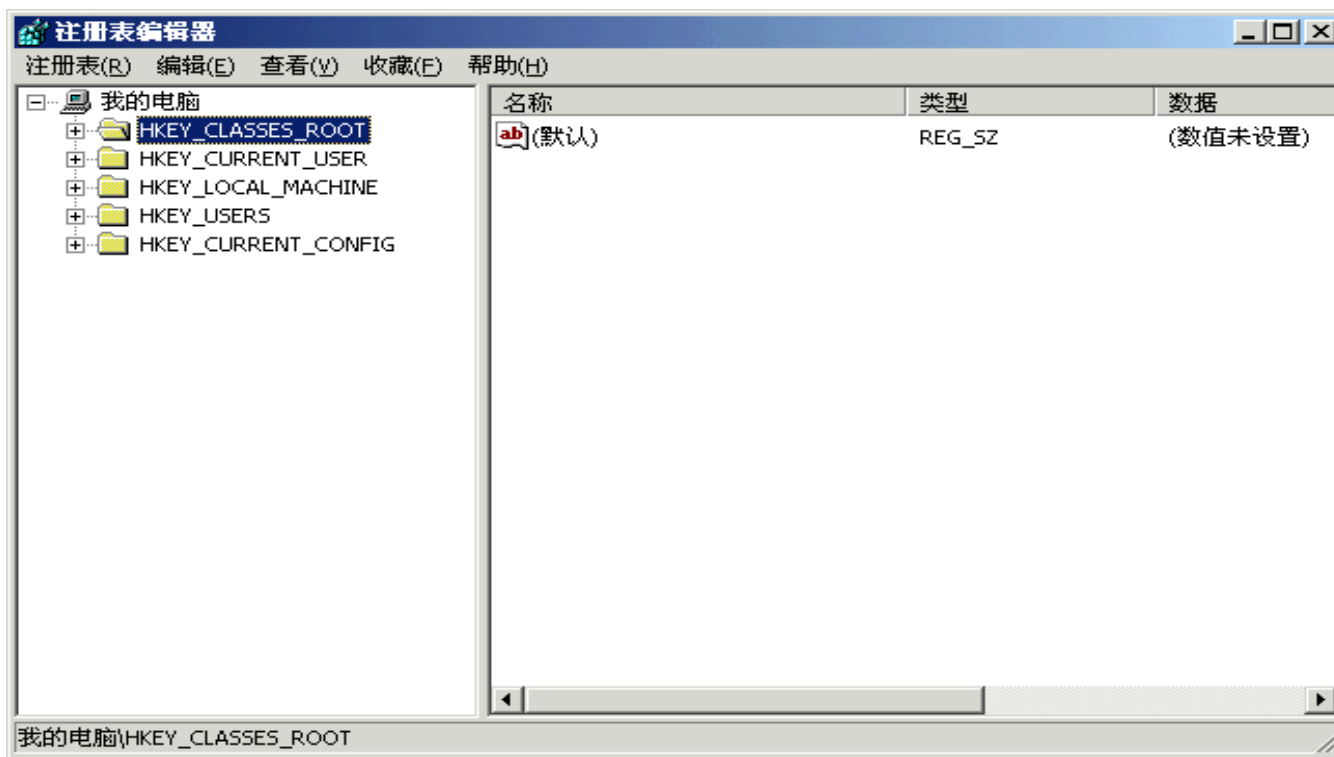
建立连接

这样可以实现对某一网站的端口扫描

3.3.2 注册表编程



- 注册表在计算机中由键名和键值组成，注册表中存储了Window操作系统的所有配置。黑客90%以上对Windows的攻击手段都离不开读写注册表。
- 在运行窗口中输入“regedit”命令可以进入注册表，注册表的界面如图所示。



3.3.2 注册表编程



- 注册表的句柄可以由调用RegOpenKeyEx()和RegCreateKeyEx()函数得到的
- 通过函数RegQueryValueEx()可以查询注册表某一项的值；通过函数RegSetValueEx()可以设置注册表某一项的值
- RegCreateKeyEx()函数和RegSetValueEx()函数的使用方法如程序proj3_12.cpp所示

案例名称：操作注册表

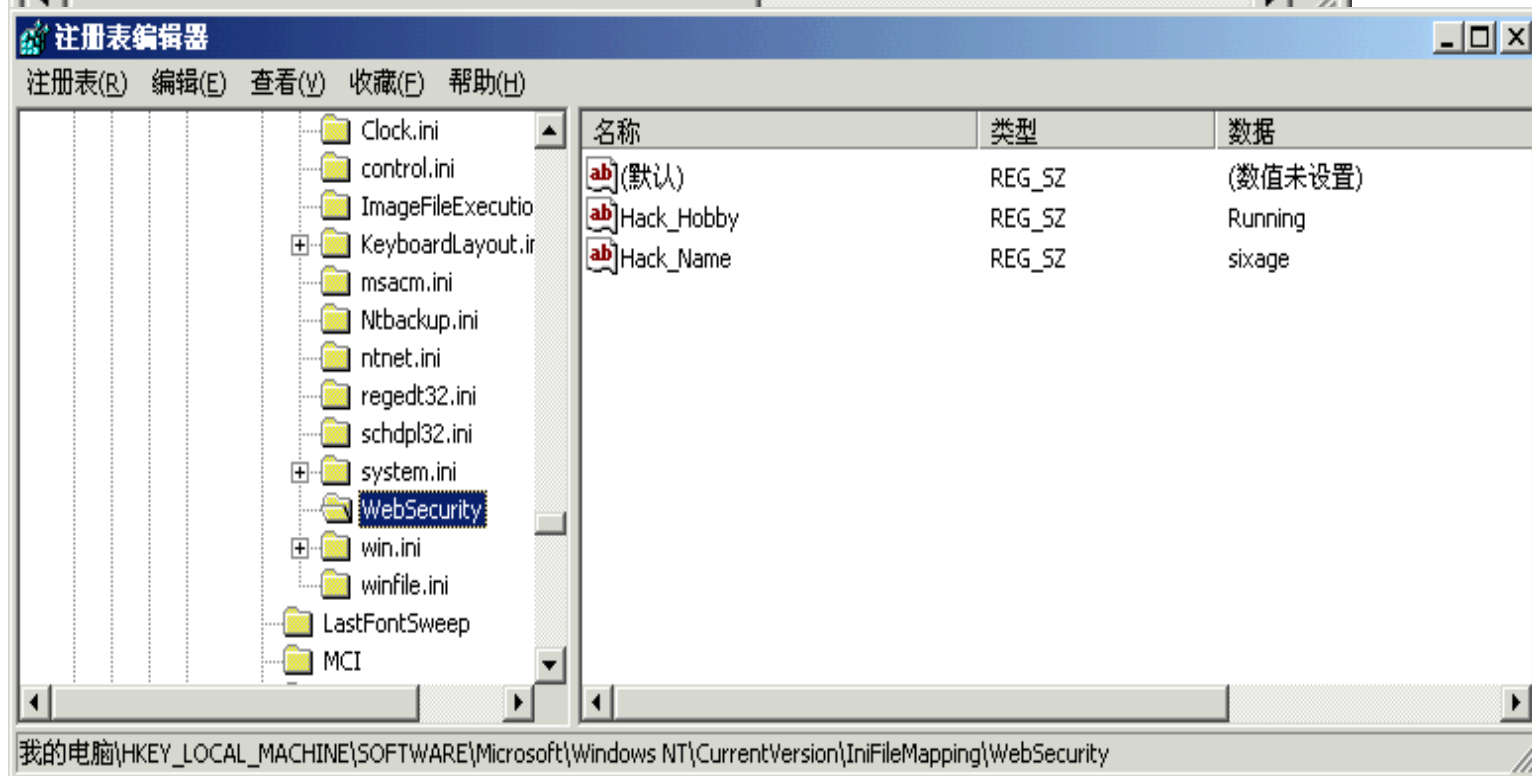
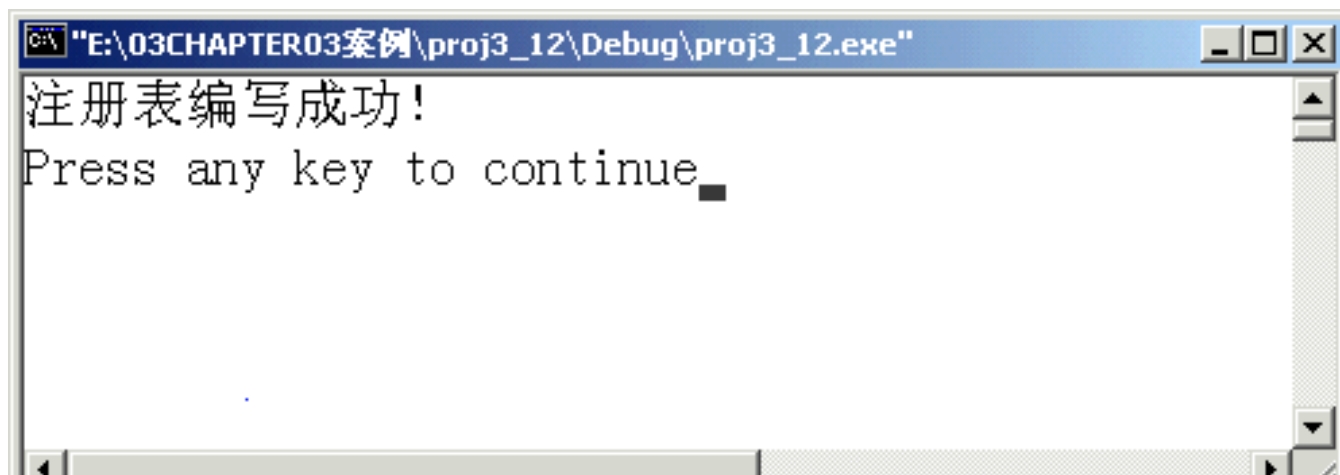


```
• #include <stdio.h>
• #include <windows.h>
•
• main()
• {
•     HKEY hKey1;
•     DWORD dwDisposition;
•     LONG lRetCode;
•     //创建
•     lRetCode = RegCreateKeyEx ( HKEY_LOCAL_MACHINE,
• "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\IniFileMapping\\WebSecurity",
•     0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE,
•     NULL, &hKey1, &dwDisposition);
•
•     //如果创建失败，显示出错信息
•     if (lRetCode != ERROR_SUCCESS){
•         printf ("Error in creating WebSecurity key\n");
•         return (0) ;
•     }
• }
```



- //设置第一个键值
- lRetCode = RegSetValueEx (hKey1, "Hack_Name",
- 0, REG_SZ,
- (byte*)"sixage",
- 100);
- //设置第二个键值
- lRetCode = RegSetValueEx (hKey1, "Hack_Hobby",
- 0, REG_SZ,
- (byte*)"Running",
- 100);
-
- //如果创建失败，显示出错信息
- if (lRetCode != ERROR_SUCCESS) {
- printf ("Error in setting Section1 value\n");
- return (0) ;
- }
- printf("注册表编写成功！ \n");
- return(0);
- }

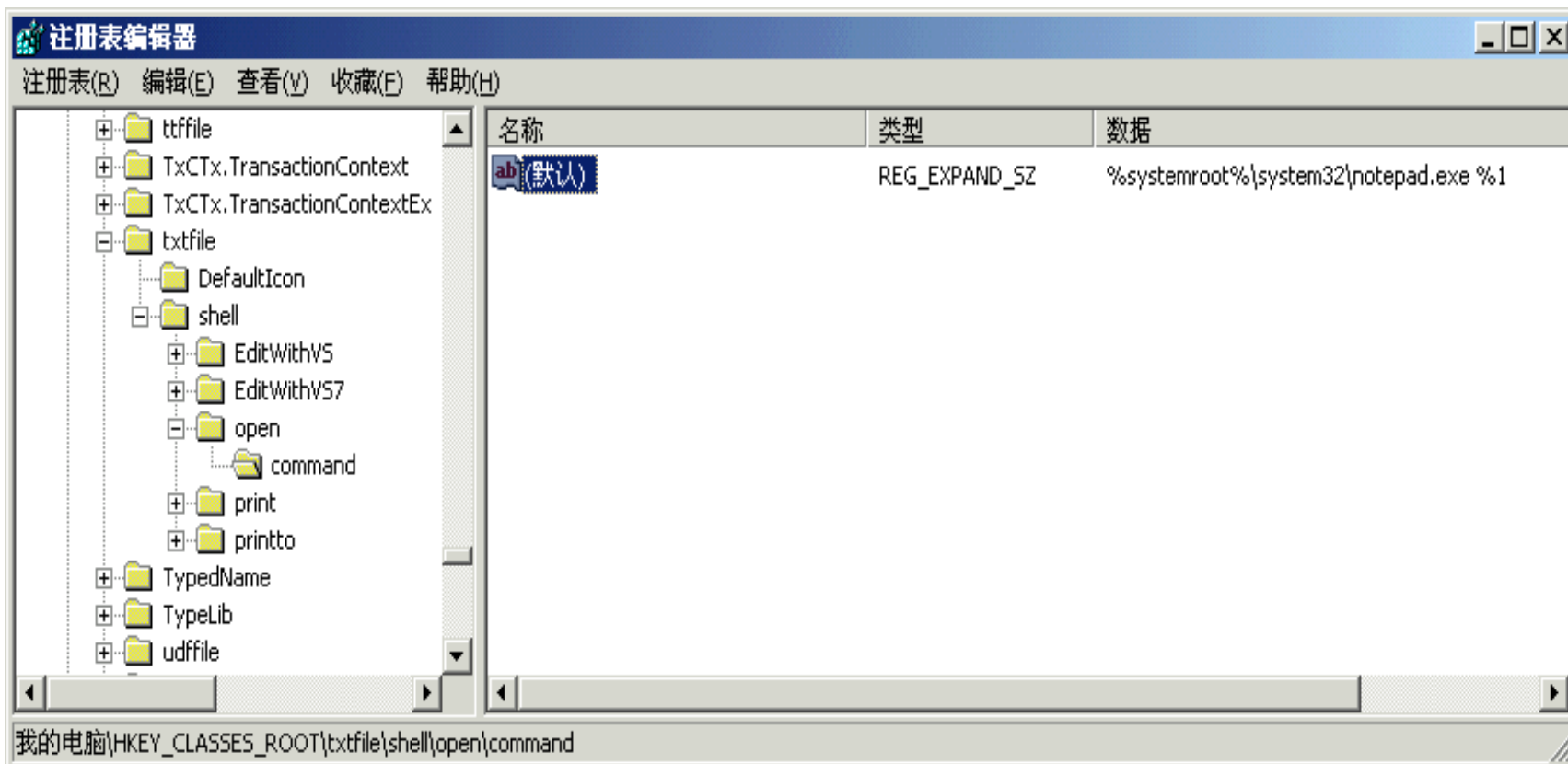
- 编译运行结果:



*判断是否中了“冰河”



- 中了“冰河”的计算机注册表都将被修改了，修改了扩展名为txt的文件的打开方式，在注册表中txt文件的打开方式定义在HKEY_CLASSES_ROOT主键下的“txtfile\shell\open\command”中，如图所示。



*判断是否中了“冰河”



- 案例名称：判断是否中了“冰河”
- 程序名称：proj3_13.cpp
-
- #include <stdio.h>
- #include <windows.h>
- main()
- {
- HKEY hKEY;
- LPCTSTR data_Set = "txtfile\\shell\\open\\command";
- long ret0 = (RegOpenKeyEx(HKEY_CLASSES_ROOT,
- data_Set, 0, KEY_READ,&hKEY));
- if(ret0 != ERROR_SUCCESS)
- //如果无法打开hKEY，则终止程序的执行
- {
- return 0;
- }
-

*判断是否中了“冰河”



- //查询有关的数据
- LPBYTE owner_Get = new BYTE[80];
- DWORD type_1 = REG_EXPAND_SZ;
- DWORD cbData_1 = 80;
- long ret1=RegQueryValueEx(hKEY, NULL, NULL,
- &type_1, owner_Get, &cbData_1);
- if(ret1!=ERROR_SUCCESS)
- {
- return 0;
- }
- if(strcmp((const char *)owner_Get,
- "%systemroot%\\system32\\notepad.exe %1") == 0)
- { printf("没有中冰河"); }
- else
- { printf("可能中了冰河"); }
- printf("\n");
- }

键名为默认
(看注册表)

- 编译执行结果:



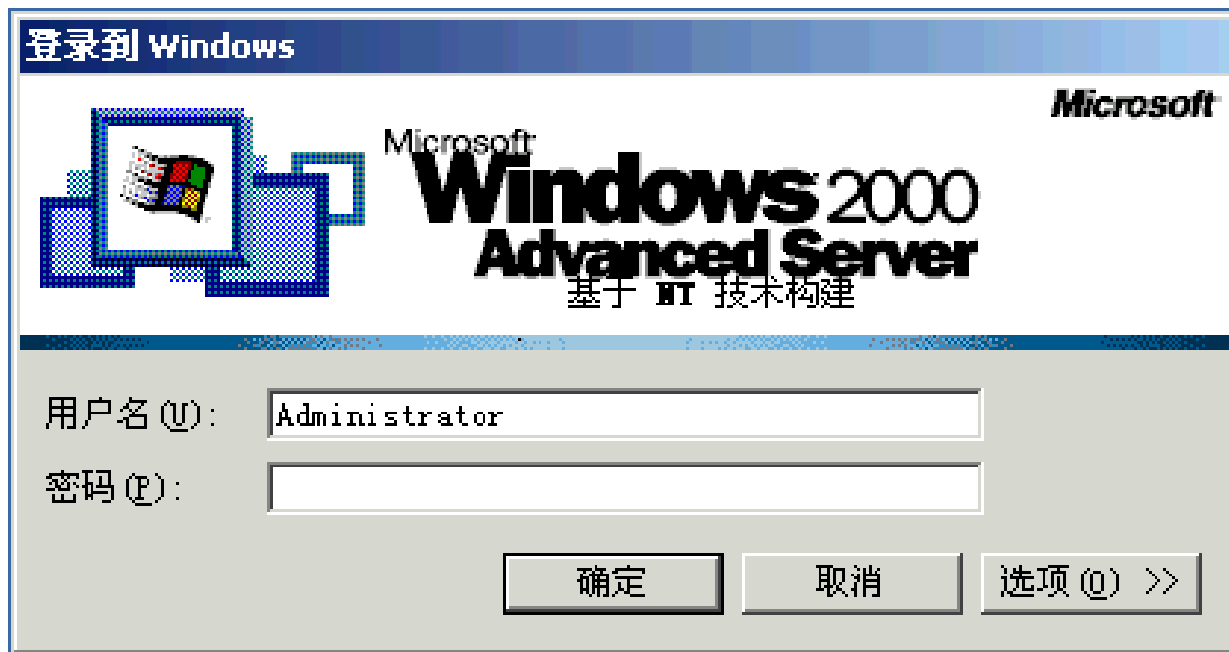
```
C:\ "E:\03CHAPTER03案例\proj3_13\Debug\proj3_13.exe"  
没有中冰河  
Press any key to continue
```

```
C:\ "E:\03CHAPTER03案例\proj3_13\Debug\proj3_13.exe"  
可能中了冰河  
Press any key to continue
```


*案例3-6 更改登录用户名



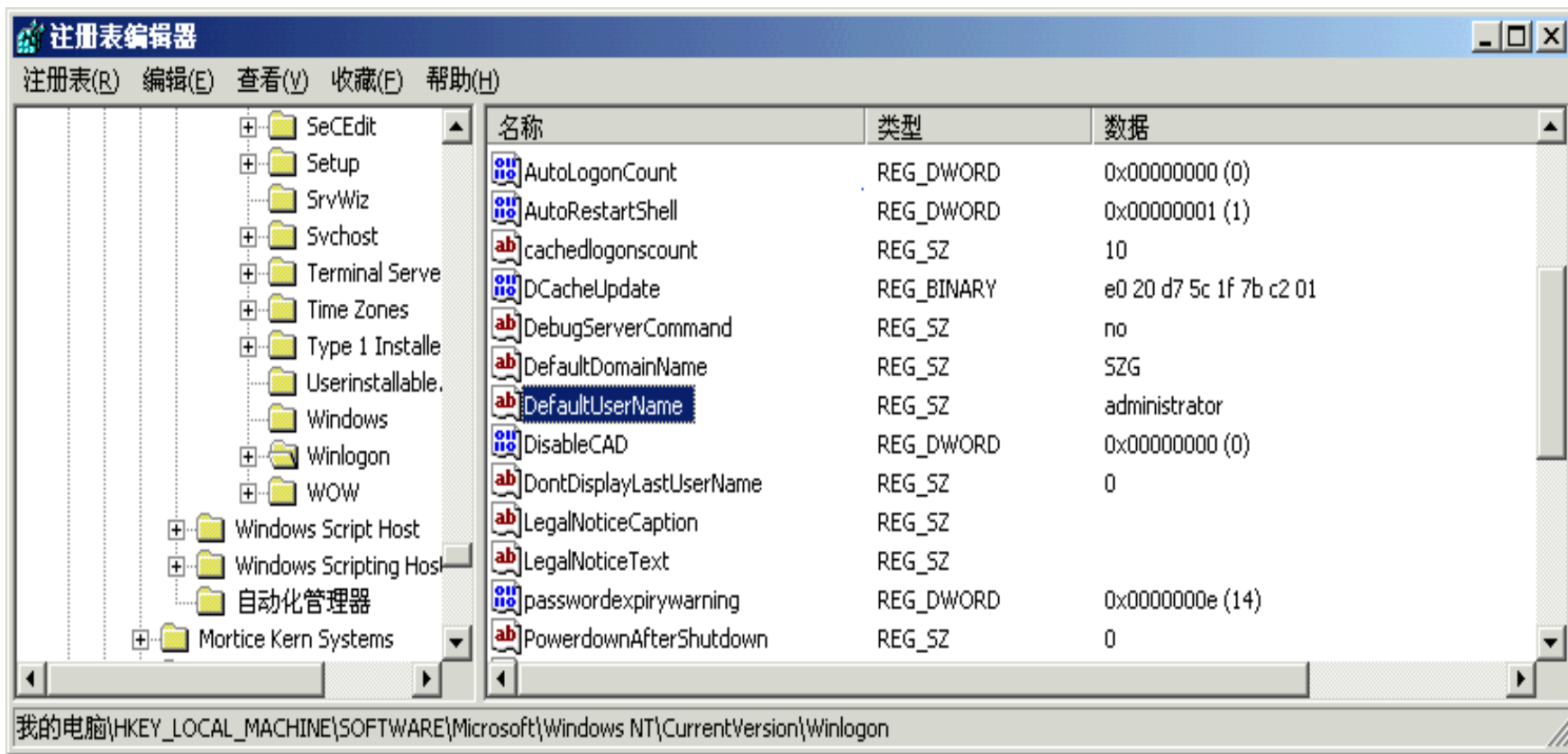
- 当用户登录系统以后，注册表中就会自动记下用户名，下次登录时再把登录名显示出来，如图所示。



案例3-6 更改登录用户名



- 当非法入侵计算机以后，同样会留下非法登录的用户名，所以需要将用户名修改回原来的值。
- 该用户名记录在注册表的HKEY_LOCAL_MACHINE主键下的SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon子键中，键的名称是：DefaultUserName，如图所示。



*程序更改系统登录用户



- 案例名称：更改系统登录用户
- 程序名称：proj3_14.cpp
-
- #include <stdio.h>
- #include <windows.h>
-
- main()
- {
- HKEY hKey1;
- LONG lRetCode;
- lRetCode = RegOpenKeyEx (HKEY_LOCAL_MACHINE,
- "SOFTWARE\\Microsoft\\Windows
- NT\\CurrentVersion\\Winlogon",
- 0, KEY_WRITE,
- &hKey1
-);
-
-
-



```
• if (lRetCode != ERROR_SUCCESS){  
•     printf ("Error in creating appname.ini key\n");  
•     return (0) ;  
• }  
•  
• lRetCode = RegSetValueEx ( hKey1,  
•     "DefaultUserName",  
•     0,  
•     REG_SZ,  
•     (byte*)"Hacker_sixage",  
•     20);  
•  
•     if (lRetCode != ERROR_SUCCESS) {  
•         printf ( "Error in setting Section1 value\n");  
•         return (0) ;  
•     }  
•     printf("已经将登录名该成Hacker_sixage");  
•     return(0);  
• }
```

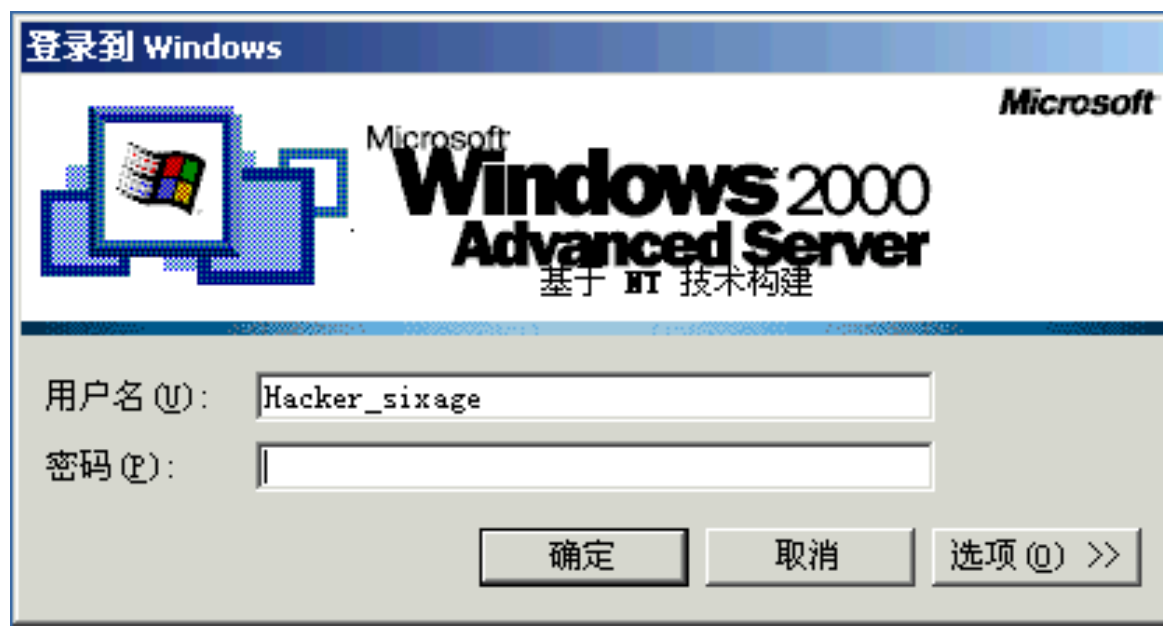
- 编译执行



```
C:\WINNT\System32\cmd.exe

C:\>proj3_14.exe
已经将登录名该成Hacker_sixage
C:\>^R_
```

- 当我们重启时,登录界面为



3.3.3 文件系统编程



- 文件系统编程非常的重要，可以在DOS命令行下执行的操作都可以使用程序实现。
- 在DOS命令行下使用命令“net user Hacker /add”添加一个用户，同样可以在程序中实现，如程序proj3_15.cpp所示
- 编译执行后可看到,已经添加用户HACKER
- 源代码如下页

```
C:\WINNT\System32\cmd.exe
C:\>net user

\\SZG 的用户帐户

-----
__vmware_user__      Administrator      ASPNET
Guest                Hacker            IUSR_WWWFOX-NET
IWAM_WWWFOX-NET      SQLDebugger      TsInternetUser
UUSR_SZG-NB
命令成功完成。
```



- `#include <stdio.h>`
- `#include <windows.h>`
- `main()`
- `{`
- `char * szCMD = "net user Hacker /add";`
- `BOOL bSuccess;`
- `PROCESS_INFORMATION piProcInfo;`
- `STARTUPINFO Info;`
- `Info.cb=sizeof(STARTUPINFO);`
- `Info.lpReserved=NULL;`
- `Info.lpDesktop=NULL;`
- `Info.lpTitle=NULL;`
- `Info.cbReserved2=0;`
- `Info.lpReserved2=NULL;`
- `bSuccess=CreateProcess(NULL,szCMD,NULL,NULL,false,NU`
`LL,NULL,NULL,&Info,&piProcInfo);`
- `if(!bSuccess) printf("创建进程失败！ ");`
- `return 1;`
- `}`

*文件拷贝和移动



- 案例名称：文件拷贝和移动
- 程序名称：proj3_16.cpp

```
#include <stdio.h>
#include <windows.h>
main()
{
```

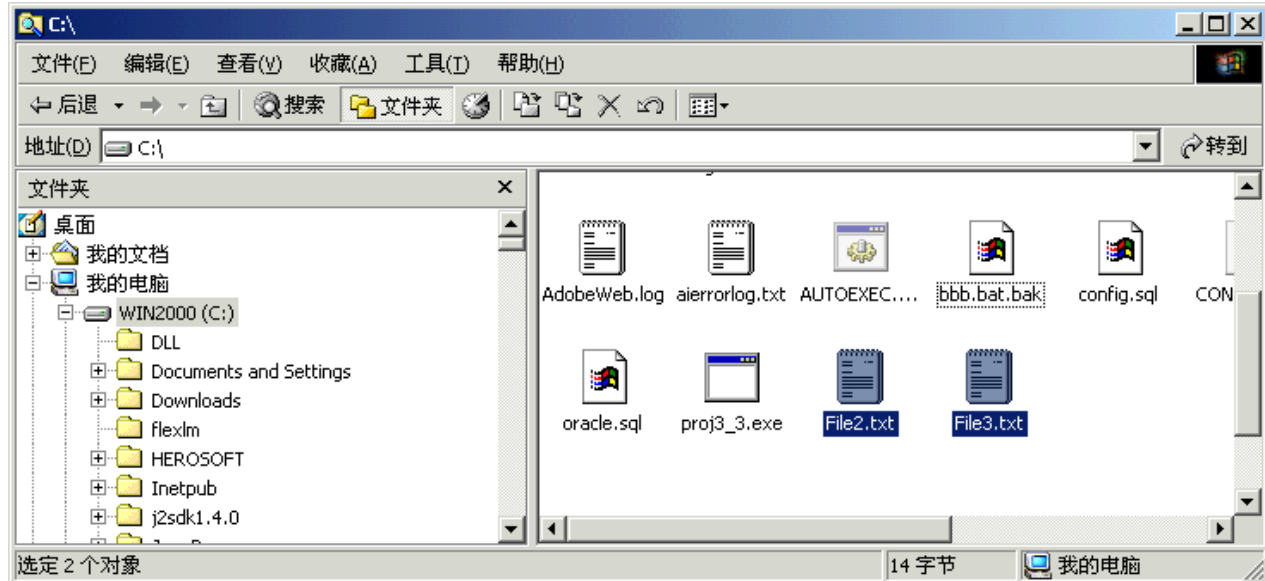
原有文件 新的文件

```
    CopyFile("C:\\File1.txt","C:\\File2.txt",TRUE);
    MoveFile("C:\\File1.txt","C:\\File3.txt");
    return 1;
}
```

- 编译执行后看到

重命名

如果已存在
file2.txt,则覆
盖



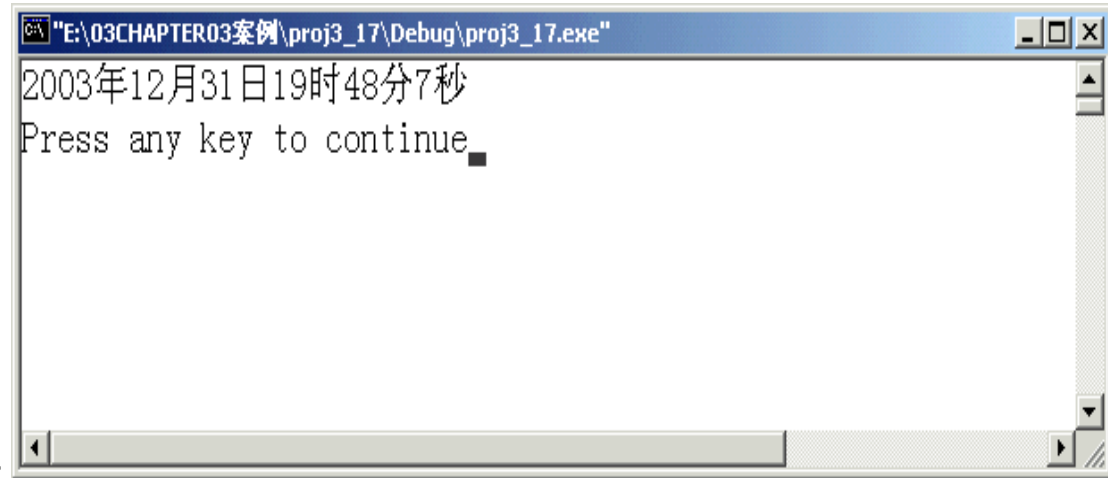
*系统时间



- 案例名称：系统时间
- 程序名称：proj3_17.cpp

- #include <windows.h>
- #include <stdio.h>
- main()
- {

- `SYSTEMTIME sysTime;`
- `GetLocalTime(&sysTime);`
- `printf("%d年%d月%d日%d时%d分%d秒\n",`
- `sysTime.wYear,sysTime.wMonth,sysTime.wDay,sysTime.wHour,`
- `sysTime.wMinute,sysTime.wSecond);`
- `return 1;`
- }
- 编译执行,如图



3.3.4 定时器编程



- 著名的“**CIH病毒**”每年定时发作，其中需要利用定时器来控制程序的执行。定时器程序分成两大类，
 - 一类是循环执行
 - 另一类是根据条件只执行一次。在程序中加载定时器，如程序proj3_18所示。

3.3.4 定时器编程



- 案例名称：定时器编程
- 程序名称：proj3_18.cpp
-
- `#include <windows.h>`
- `WNDCLASS wc;`
- `HWND h_wnd;`
- `MSG msg;`
-
- `/* 消息处理函数wndProc的声明*/`
- `long WINAPI`
`WindowProc(HWND,UINT,WPARAM,LPARAM);`
-
- `/* winMain 函数的声明*/`
- `int PASCAL WinMain(HINSTANCE h_CurInstance,`
- `HINSTANCE h_PrevInstance,LPSTR p_CmdLine,int m_Show)`
- `{`

3.3.4定时器编程



- `/*初始化wndclass结构变量*/`
- `wc.lpfWndProc = WindowProc;`
- `wc.hInstance = h_CurInstance;`
- `wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);`
- `wc.lpszClassName = "TheMainClass";`
-
- `/* 注册WndClass结构变量*/`
- `RegisterClass(&wc);`
- `/* 创建窗口*/`
- `h_wnd=CreateWindow("TheMainClass","Our first Window",WS_OVERLAPPEDWINDOW,0,0,400,500,0,0,h_CurInstance,0);`
- `/* 显示窗口*/`
- `ShowWindow(h_wnd,SW_SHOWMAXIMIZED);`
- `/*消息循环*/`
- `while(GetMessage(&msg,NULL,0,0)) DispatchMessage(&msg);`
- `return (msg.wParam);`
- `}`

3.3.4定时器编程



- #define ID_TIMER 1
- /* 定义消息处理函数*/
- long WINAPI WindowProc(HWND h_wnd,UINT WinMsg, WPARAM w_param,LPARAM l_param)
- {
- static BOOL fFlipFlop = FALSE ;
- HBRUSH hBrush ;
- HDC hdc ;
- PAINTSTRUCT ps ;
- RECT rc ;

3.3.4定时器编程



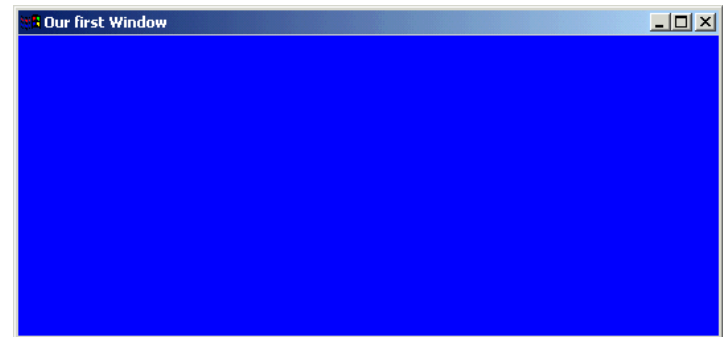
- switch (WinMsg)
- {
- case WM_CREATE:
 - SetTimer (h_wnd, ID_TIMER, 1000, NULL) ;
 - return 0 ;
- case WM_TIMER :
- MessageBeep (-1) ;
- fFlipFlop = !fFlipFlop ;
- **触发** InvalidateRect (h_wnd, NULL, FALSE) ;
return 0 ;
- case WM_PAINT :
- hdc = BeginPaint (h_wnd, &ps) ;
- GetClientRect (h_wnd, &rc) ;
- hBrush = CreateSolidBrush (fFlipFlop ? RGB(255,0,0) : RGB(0,0,255)) ;

设置定时
器

3.3.4定时器编程



- FillRect (hdc, &rc, hBrush) ;
- EndPaint (h_wnd, &ps) ;
- DeleteObject (hBrush) ;
- return 0 ;
-
-
- case WM_DESTROY :
 - KillTimer (h_wnd, ID_TIMER) ;
 - PostQuitMessage (0) ;
 - return 0 ;
- }
- return DefWindowProc(h_wnd,WinMsg,w_param,l_param);
- }
- 编译执行出现红蓝交替显示窗口



3.3.5 驻留程序编程

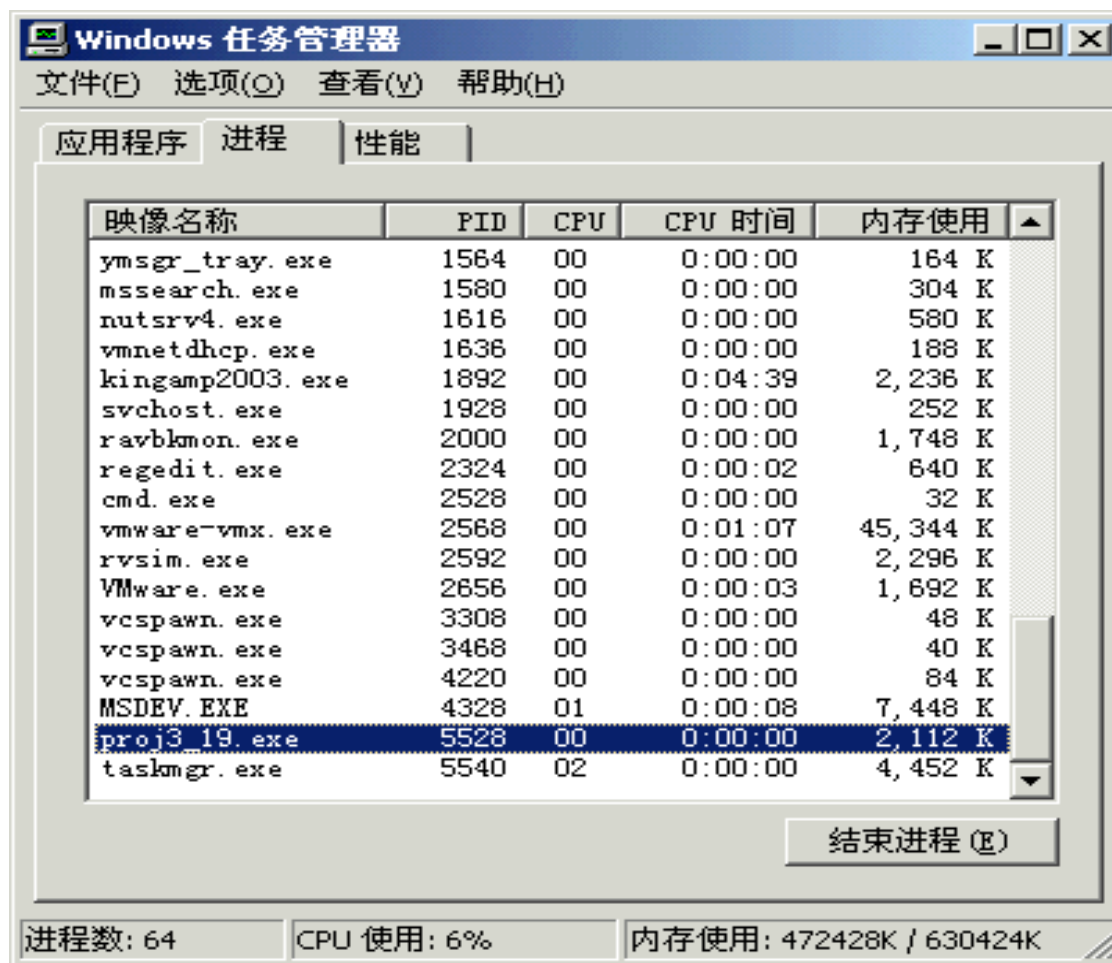


- 一般程序运行时都有窗口
- 一般后门或病毒程序是后台运行的（即驻留程序）
- 其实编写驻留程序很方便，在上述proj3_18.cpp中将ShowWindow()函数的“SW_SHOWMAXIMIZED”改为“SW_HIDE”即可。
- 可参看proj3_19.cpp中
 ShowWindow(h_wnd, SW_HIDE);

3.3.5 驻留程序编程



编译执行没有任何显示，打开任务管理器，可以看到
proj3_19.exe正在运行



3.3.5 驻留程序编程

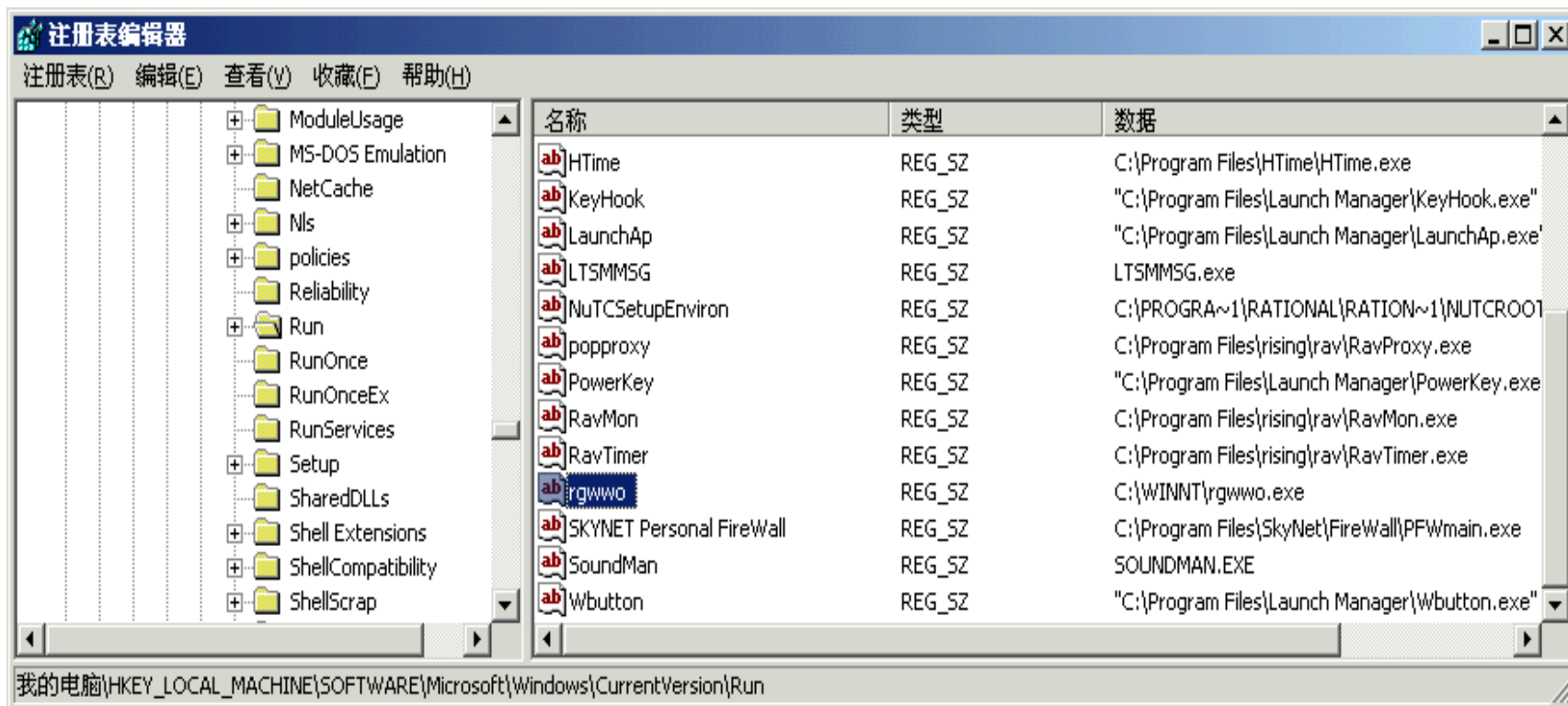


- 程序运行时不显示界面，为了实现自动驻留，一般有两种方法
 - 更改注册表启动项
 - 让该程序与用户的某一操作关联
- 比如：“冰河”木马采用第二种方法
 - 当用户双击扩展名为txt的文本文件时，自动加载“冰河”程序

案例3-7 “冰河” 原型



- 第一种方法实现起来比较简单，注册表的自启动项的键值在“HKEY_LOCAL_MACHINE”主键下的“SOFTWARE\Microsoft\Windows\CurrentVersion\Run”子键中，如图所示（可手工可程序添加）



案例3-7 “冰河” 原型



- 第二种方法的实现是使用“HKEY_CLASSES_ROOT”主键下“txtfile\shell\open\command”键
- 程序实现的**功能**是：当用户双击打开一个文本文件时,先启动要驻留的程序,然后再启动记事本打开这个文本文件。
- **关键**：用户双击的文本文件地址如何通过驻留程序传递给记事本。

案例3-7 “冰河” 原型



- 第一步，先修改注册表关联（可手工可程序）
 - 在“HKEY_CLASSES_ROOT”主键下
“txtfile\shell\open\command”键值改为驻留程序（e:\proj3_20.exe %1）
- 第二步就是实现关键部分，关键部分实现方法如proj3_20.cpp程序

案例3-7 “冰河” 原型



- proj3_20.cpp源程序如下：
- #include <windows.h>
- WNDCLASS wc;
- HWND h_wnd;
- MSG msg;
- /* 消息处理函数wndProc的声明*/
- long WINAPI WindowProc(HWND,UINT,WPARAM,LPARAM);
- /* winMain 函数的声明*/
- int PASCAL WinMain(HINSTANCE h_CurInstance,
- HINSTANCE
- h_PrevInstance,LPSTR p_CmdLine,int m_Show)
- {

案例3-7 “冰河” 原型



- `BOOL bSuccess;`
- `PROCESS_INFORMATION piProcInfo;`
- `STARTUPINFO Info;`
- `Info.cb = sizeof(STARTUPINFO);`
- `Info.lpReserved = NULL;`
- `Info.lpDesktop = NULL;`
- `Info.lpTitle = NULL;`
- `Info.cbReserved2 = 0;`
- `Info.lpReserved2 = NULL;`
- `char lpAppName[100];`
- `strcpy(lpAppName, "notepad.exe");`

建立一个可以
执行DOS命令
的对象

案例3-7 “冰河” 原型



- if(strcmp(p_CmdLine, "") != 0)
- strcat(lpAppName, p_CmdLine);
- bSuccess = CreateProcess(NULL, lpAppName, NULL, NULL, false, NULL, NULL, NULL, &Info, &piProcInfo);

实现记事本
打开文件

- /*初始化wndclass结构变量*/
- wc.lpfnWndProc = WindowProc;
- wc.hInstance = h_CurInstance;
- wc.hbrBackground
= (HBRUSH)GetStockObject(WHITE_BRUSH);
- wc.lpszClassName = "TheMainClass";

以下就是自
己驻留程序
的功能实现

案例3-7 “冰河” 原型



/* 注册WndClass结构变量*/

- RegisterClass(&wc);

/* 创建窗口*/

```
h_wnd=CreateWindow("TheMainClass","Our first Window",  
WS_OVERLAPPEDWINDOW,0,0,400,500,0,0,h_CurInstan  
ce,0);
```

/* 显示窗口*/

- ShowWindow(h_wnd,SW_HIDE);

/*消息循环*/

```
while(GetMessage(&msg,NULL,0,0))
```

```
    DispatchMessage(&msg);
```

- return (msg.wParam);
- }

案例3-7 “冰河” 原型



- /* 定义消息处理函数*/
- long WINAPI WindowProc(HWND h_wnd,UINT WinMsg, WPARAM w_param,LPARAM l_param)
- {
 - switch (WinMsg)
 - {
 - case WM_DESTROY :
 - PostQuitMessage (0) ;
 - return 0 ;
 - }
 - return
- DefWindowProc(h_wnd,WinMsg,w_param,l_param);
- }

3.3.6 多线程编程



- 用多线程技术编程有两大优点：
 - 1、提高CPU的利用率
 - 2、采用多线程技术，可以设置每个线程的优先级，调整工作的进度。
- 在实际开发过程中，一定要有一个主进程，其他线程可以共享该进程也可以独立运行，每个线程占用CPU的时间有限制，可以设置运行优先级别。

*独立线程程序的编写



- 案例名称：独立线程程序的编写
- 程序名称：proj3_21.cpp
-
- `#include <process.h>`
- `#include <stdlib.h>`
- `#include <stdio.h>`
-
- `int addem(int);`
- `int main(int argc, char *argv[])`
- `{`
- `_beginthread((void (*)(void *))addem, 0, (void *)10);`
- `_beginthread((void (*)(void *))addem, 0, (void *)11);`
- `addem(12);`
- `return 0;`
- `}`
-

*独立线程程序的编写



```
• int addem(int count)
• {
•     int    i;
•     long sum;
•
•     sum = 0;
•     for (i=0; i<=count; ++i) {
•         printf("The value of %d is %d\n", count, i);
•         sum += i;
•     }
•     printf("The sum is %d\n", sum);
•     return 0;
• }
```

*独立线程程序的编写



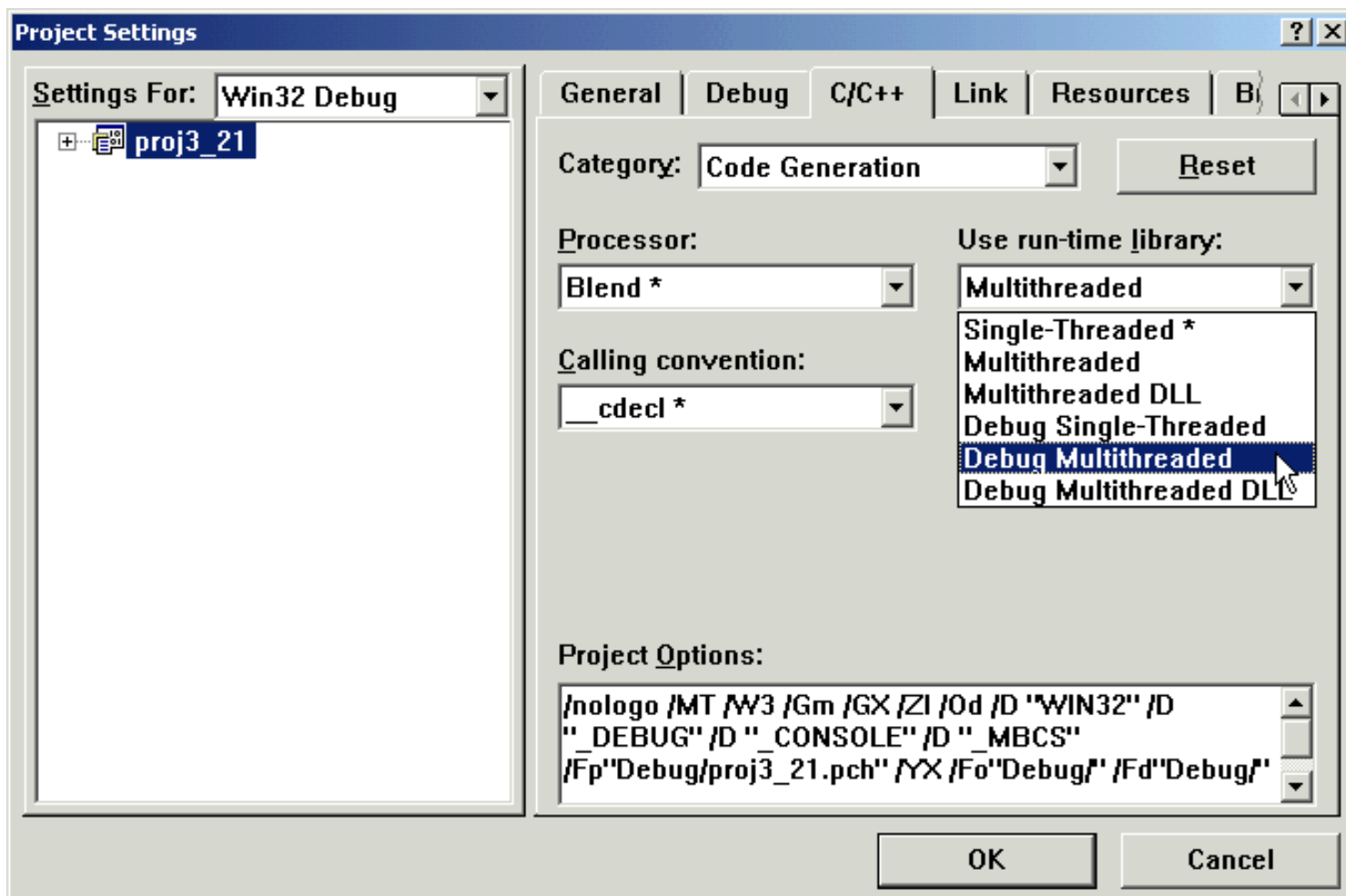
- 编译执行，出错

The screenshot shows the Microsoft Visual C++ IDE. The main window displays a C++ program named 'proj3_21.cpp'. The code includes headers for <process.h>, <stdlib.h>, and <stdio.h>. It defines a function 'addem(int)' and a 'main' function. In the 'main' function, two threads are created using '_beginthread' to call 'addem' with arguments 10 and 11. After the threads, 'addem(12)' is called, and the program returns 0. The left sidebar shows the 'proj3_21 classes' and 'FileView' tabs. The bottom status bar indicates 'Ln 4, Col 51'. The output window at the bottom shows the compilation process and an error message: 'error C2065: '_beginthread' : undeclared identifier'. The error message is: 'e:\03chapter03案例\proj3_21\proj3_21.cpp(8) : error C2065: '_beginthread' : undeclared identifier Error executing cl.exe. proj3_21.exe - 1 error(s), 0 warning(s)'. The output window also shows 'Configuration: proj3_21 - Win32 Debug' and 'Compiling... proj3_21.cpp'. The status bar at the bottom shows 'Build', 'Debug', 'Find in Files 1', 'Find in Files 2', 'SQL Debug', and 'Ln 4, Col 51'.

*独立线程程序的编写



- 由于基于控制台程序默认单线程执行，所以修改



*独立线程程序的编写



- 再编译执行得到如下结果

```
"E:\03Chapter03案例\proj3_21\Debug\proj3_21.exe"
The value of 11 is 5
The value of 10 is 6
The value of 11 is 6
The value of 10 is 7
The value of 11 is 7
The value of 10 is 8
The value of 11 is 8
The value of 10 is 9
The value of 11 is 9
The value of 10 is 10
The value of 11 is 10
The sum is 55
The value of 11 is 11
The sum is 66
The sum is 78
Press any key to continue
```


*多个线程共享参数



- 案例名称：多个线程共享参数
- 程序名称：proj3_22.cpp
-
- `#include <process.h>`
- `#include <stdlib.h>`
- `#include <stdio.h>`
- `int addem(int);`
- `int x; //全局变量`
-
- `int main(int argc, char *argv[])`
- `{`
- `x=0;`
- `_beginthread((void (*)(void *))addem, 0, (void *)1);`
- `_beginthread((void (*)(void *))addem, 0, (void *)2);`
- `addem(3);`
- `return 0;`
- `}`

*多个线程共享参数



```
• int addem(int index)
• {
•     while (x <= 50){
•         x = x+1;
•         printf("%d: %d\n", index, x);
•     }
•     return 0;
• }
```

A screenshot of a Windows command prompt window. The title bar reads "E:\03CHAPTER03案例\proj3_22\Debug\proj3_22.exe". The window contains the following output:

```
2: 37
1: 38
2: 39
1: 40
2: 41
1: 42
2: 43
1: 44
2: 45
1: 46
2: 47
3: 48
1: 49
3: 50
1: 51
Press any key to continue.
```

本章总结



- 本章需要重点掌握**Windows**操作系统的内部机制，理解**C**语言四个阶段编程的特点。
- 重点掌握网络安全编程领域的**Socket**编程、注册表编程、驻留程序的编程和多线程编程。

本章习题



- **【1】**、简述Windows操作系统的内部机制。
- **【2】**、简述学习Windows下编程的注意点。
- **【3】**、比较C语言四个发展阶段编程的特点。
- **【4】**、用程序说明MFC的事件处理机制。
- **【5】**、编写程序实现功能：清除“冰河”程序和文本文件的关联。
（上机完成）
- **【6】**、编写程序实现功能：在每天夜里十二点，自动删除C盘下的File4.txt文件。（上机完成）
- **【7】**、编写程序实现功能：当登录系统以后，自动执行一个程序，该程序将系统登录名改成Administrator。（上机完成）
- **【8】**、编写程序实现功能：当用户用鼠标双击一个文本文件的时候，自动删除该文件。（上机完成）