WinSock网络编程

1. 概述

80's初，ARPA(美国国防部高级研究计划局)
®加利福尼亚大学Berkeley分校提供资金，®开发在UNIX下实现TCP/IP协议。
®为TCP/IP开发了一个API — Socket接口（套接口）— 俗称Bekeley套接口模型。

90's初，Microsoft等公司
®基于Bekeley套接口模型
®制定了Windows Sockets规范(简称WinSock)
®已是TCP/IP网络的标准。

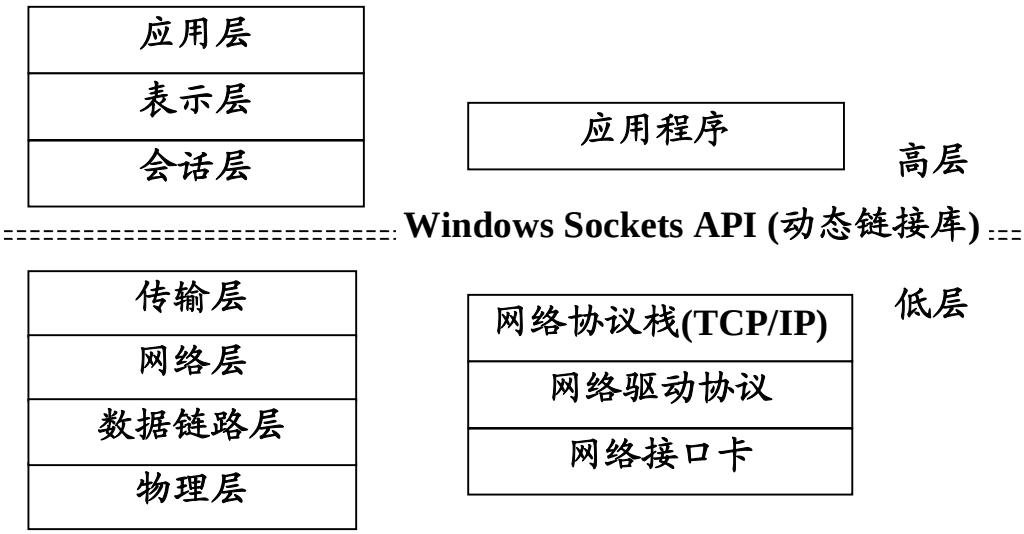
1993.1，v1.1
1995.5，v2.0，增加了QOS (网络服务质量控制)

80年代初，美国国防部高级研究计划局(ARPA)给加利福尼亚大学Berkeley分校提供了资金，让他们在UNIX操作系统下实现TCP/IP协议。在这个项目中，研究人员为TCP/IP网络通信开发了一个API（应用程序接口）。这个API称为Socket接口（套接字）。今天，SOCKET接口是TCP/IP网络最为通用的API，也是在INTERNET上进行应用开发最为通用的API。

90年代初，由Microsoft联合了其他几家公司共同制定了一套WINDOWS下的网络编程接口，即WindowsSockets规范。它是BerkeleySockets的重要扩充，主要是增加了一些异步函数，并增加了符合Windows消息驱动特性的网络事件异步选择机制。WINDOWS SOCKETS规范是一套开放的、支持多种协议的Windows下的网络编程接口。从1991年的1.0版到1995年的2.0.8版，经过不断完善并在Intel、Microsoft、Sun、SGI、Informix、Novell等公司的全力支持下，已成为Windows网络编程的事实上的标准。

2. WinSock模型

提供TCP/IP传输层的接口：



①TCP(传输控制协议)

提供虚电路和面向连接的数据流传输服务。
实现无差错无重复的顺序数据传输。

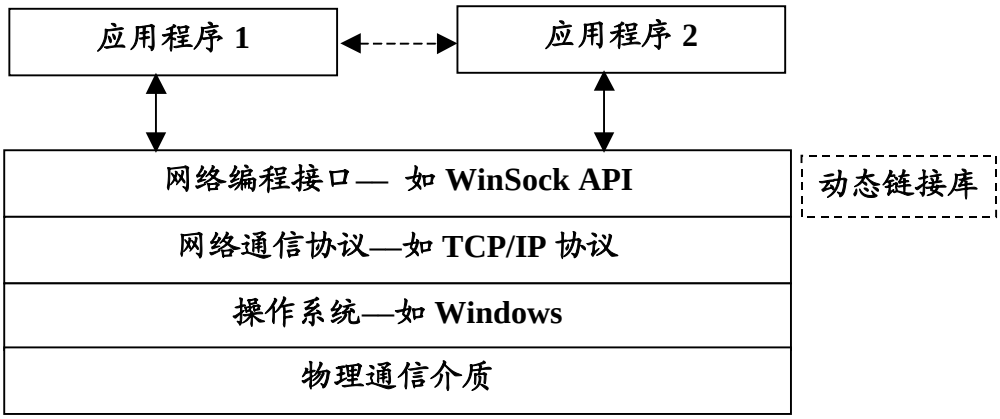
②UDP(用户数据报协议)

提供无连接的数据报传输服务。

数据通过相互独立的报文进行传输，是无序的，并且不保证可靠、无差错。

3. WinSock DLL

•WinSock与操作系统的关系



•动态链接库：

16位版：WINSOCK.DLL 32位版：WSOCK32.DLL

①DLL装载

WinSock服务由动态连接库WinSock DLL提供，

它完成WinSock的初始化任务，协商WinSock 的版本支持，并分配必要的资源。

在使用WinSock API之前，必须调用:

· `int WSAStartup(WORD v, (LPWSADATA)&WD)`

其中:

v —— 指示应用程序对WinSock版本的要求，
低字节为主版本号，高字节为副版本号。
例：**v1.1 ® v=Ox0101, v2.0 ® v=Ox0002,**

WD—返回WinSock的实现信息。

WD是一个WSADATA结构:

```
struct  WSADATA{
   WORD    wVersion;
   WORD    wHighVersion;
   char    szDescription[WSADESCRIPTION_LEN+1];
   char    szSystemStatus[WSASYSSTATUS_LEN+1];
```

```
    unsigned short iMaxSockets;
    unsigned short iMaxUdpDg;
    char FAR *lpVendorInfo;
};
```

结构成员说明

wVersion:	**DLL**支持的**WinSock**规范的版本;
wHighVersion:	**DLL**可支持的**WinSock**规范的最高版本;
szDescription:	**DLL**的说明及厂商描述;
szSystemStatus:	**DLL**将相关的状态和配置信息;
iMaxSockets:	一个进程可以打开的最大套接口数目;
iMaxUdpDg:	应用程序发送或接收的最大**UDP**数据报的大小; 如果应用程序没有给出限制,**iMaxUdpDg**为**0**(隐含为**8192**字节)。最小值为**512**。
lpVendorInfo:	指向厂商规定数据结构的远指针。

调用成功,返回**0**。

②DLL卸载

当不需**WinSock DLL**的服务,释放**DLL**所使用的资源。

应用程序必须调用:

· **int WSACleanup()**

调用成功,返回**0**。

对应于每一次**WSAStartup()**调用必须有一个**WSACleanup()**调用。

4. 套接口Socket

Socket 实际上是一个通信端口; 一个**Socket**是通讯的一端。

网络通信将通过各自的**Socket**相联系。

在应用开发中就像使用文件句柄一样,

应用程序向操作系统申请,

由操作系统分配**本地唯一**的**Socket**端口号。

然后,可以对**Socket**句柄进行读,写操作。

·创建**Socket**:

```
SOCKET socket(
    int af,          //套接口所用地址族
    int type,        //套接口类型
```

```
    int protocol      //套接口所用协议
   )
```

	参数	说明
①af	AF_INET	TCP/IP地址
	AF_UNIX	UNIX地址
	... ...	
②type	SOCK_STREAM	数据流套接口，对应TCP协议
	SOCK_DGRAM	数据报套接口，对应UDP协议
	... ...	……
③protocol	IPPROC_TCP	使用TCP/IP的TCP协议
	IPPROC_UDP	使用TCP/IP的UDP协议
	... ...	……
	0	①和②基本确定了一种协议，若调用者不想指定，设置为0。

返回值：

若无错误发生，socket()返回引用套接口的描述字(套接口号)。

否则的话，返回SOCKET_ERROR错误，即-1。

应用程序可通过WSAGetLastError()获取相应错误代码。

5. 主机地址标识

网络环境中的唯一通信端点标识。

包含：协议、IP地址、端口。（俗称三元组）

关于端口：

在TCP/IP中，TCP与UDP使用彼此独立的端口；

端口大小：16bit(共2^{16}个)

端口分为：

①系统全局端口：1~1023;

例，HTTP为TCP/80，FTP为TCP/21、UDP/69，SMTP为TCP/25

②系统自动分配端口：1024~5000;

③自由端口：5000~65535;

6. 主机地址标识的数据结构

```
struct  sockaddr {
   u_short sa_family;   //协议族
   char sa_data[14];    //主机地址标识(端口号、IP地址)
};
```

```
struct  sockaddr_in{
   short   sin_family;        //协议族
   u_short sin_port;          //16bit端口号，网络字节顺序
   struct in_addr sin_addr;//32bit的IP地址，网络字节顺序
   char    sin_zero[8];       //未用
};
```

其中:

```
struct  in_addr{
   u_long  s_addr;          //32bit的IP地址，网络字节顺序
};
```

网络字节顺序: 16 bit/32 bit整数存放格式—高字节在前，低字节在后。

•设置主机地址

```
//-------------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family        = AF_INET;        //TCP/IP协议
   A->sin_port          = htons(Port);    //端口号。
   A->sin_addr.s_addr   = inet_addr(IP);  //IP地址。
}
//-------------------------------------------------
```

函 数	作 用
htons()	把16 bit的数字从主机字节顺序转换到网络字节顺序
inet_addr()	把一个IP地址格式"A.B.C.D"转换成32 bit的网络字节顺序

注: Intel CPU的主机字节顺序: 16 bit/32 bit整数存放格式—低字节在前，高字节在后。

7. Socket号与主机地址捆绑

将IP地址和端口号与所创建的Socket号联系起来。

```
·int bind(
SOCKET s,                       //待捆绑Socket
struct sockaddr far *name,      //赋予Socket的主机地址标识
int len                         // name的长度
);
```

调用成功，返回0。

8. WinSock操作模式

①同步模式或阻塞模式(blocking mode)

采用DOS技术编程，某些WinSock函数(同步函数)直到完成操作后才返回。

例，当执行数据接收函数revc()时，一直等待对方发送数据，直到接收到数据后才返回。

②异步模式或非阻塞模式(non-blocking mode)

采用Windows技术编程，利用消息(事件驱动)的特点，使同步函数变为异步函数(不产生阻塞)。

关键: 异步选择函数WSAAsyncSelect()的使用。

WSAAsyncSelect()可设置一个或多个网络事件消息，

如，已收到数据、数据发送完毕、客户机请求连接、服务器已完成连接等网络事件。

当设置的网络事件发生时，Windows应用程序的窗口函数将收到一个消息。通过这个消息就可以进行相应的处理。

```
int WSAAsyncSelect(
SOCKET s,              //需要事件驱动的套接口
HWND hWnd,             //接收消息的窗口句柄
unsigned int wMsg,     //网络事件发生时的消息字
long lEvent            //用于指明感兴趣的网络事件集合
);
```

lEvent参数由下表中列出的值组成:

值	意义
FD_READ	已接收到数据
FD_WRITE	数据发送完毕
FD_OOB	已接收到边带数据
FD_ACCEPT	客户机请求连接，用于服务器端
FD_CONNECT	服务器已完成连接，用于客户机端
FD_CLOSE	连接关闭(对方的套接口关闭)

例:

```
WSAAsyncSelect(S,hW,WM_USER+1,
               FD_ACCEPT|FD_READ|FD_CLOSE);
```

程序结束时，应注销异步选择:

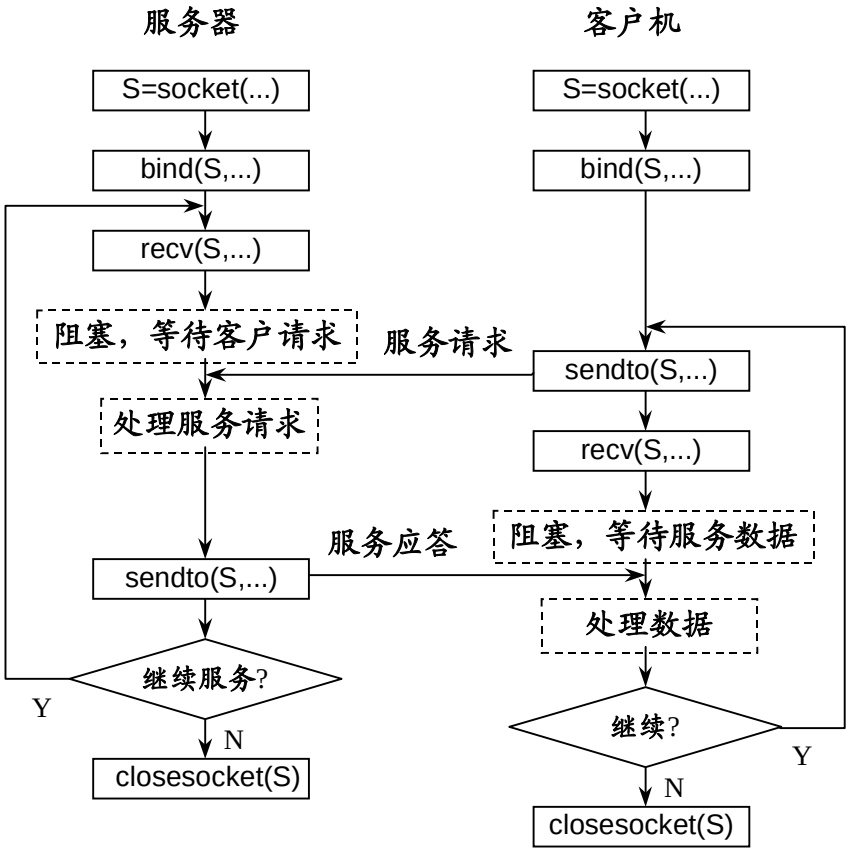
```
WSAAsyncSelect(S, hW, 0, 0);
```

9. 无连接协议的同步模式编程

无连接服务器一般都是面向事务处理的。

一个请求一个应答就完成了客户程序与服务程序之间的相互作用。

①工作过程:



无连接套接口应用程序时序图

服务器首先启动，通过调用**socket()**建立一个套接口，
然后**bind()**将该套接口和本地地址(**IP**地址和端口)联系在一起，
服务器调用**recv()**等待接收数据。

客户机通过调用**socket()**建立一个套接口，
然后**bind()**将该套接口和本地地址(**IP**地址和端口)联系在一起，
客户机调用**sendto()**向服务器发送数据;

服务器的**recv()**接收到客户机的数据后，
调用**sendto()**向客户机发送应答数据;

客户机的**recv()**便接收到了服务器的应答数据;

最后，待数据传送结束后，双方调用**closesocket()**关闭套接口。

②编程示例：

```
// UDP (TCP/IP) for the console application.
//
//VC6.0 add WSOCK32.LIB in Project->Settings...->Link

#include "stdafx.h"
#include <winsock.h>          //by user
#include <stdlib.h>           //by user

WORD RPort = 6666;            //远程端口RemotePort
char RIP[16]="127.0.0.1";     //远程IP地址RemoteIPAddr
WORD LPort = 7777;            //本地端口LocalPort
char LIP[16]="127.0.0.1";     //本地IP地址LocalIPAddr

SOCKET  S;                    //套接口SOCKET
struct sockaddr_in rAddr;     //远程参数，remoteAddr
struct sockaddr_in lAddr;     //本地参数，localAddr

WSADATA  WD;                  //WinSock DLL信息
int r;                        //result;
//-----------------------------------------------
void ShowInfo(char *info)
{   puts(info);   exit(1);  }
//-----------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family = AF_INET;           //TCP/IP协议
   A->sin_port = htons(Port);         //端口。
   A->sin_addr.s_addr =inet_addr(IP); //IP网址。
}
//-----------------------------------------------
void main()
{
WORD v;                   //wVersionRequested;
//-- - - - - - - - Startup Win Socket - - - - - - - -
   v=0x0101;              //0x0101 for v1.1, 0x0002 forv 2.0
   r = WSAStartup(v, (LPWSADATA)&WD);
   if(r != 0)  ShowInfo("Start_Error");
//-- - - - - - - - Create Win Socket - - - - - - - -
   S = socket(PF_INET, SOCK_DGRAM, 0);
   if(S == -1)  ShowInfo("Socket_Create_Error");

int l=sizeof(rAddr);
char Msg[80];

   puts("Type exit then Quit Program!");

   SetSockAddr(lAddr, LPort, LIP);
```

```
   r = bind(S,(struct sockaddr far *)&lAddr, sizeof(lAddr));
   if(r == -1)  ShowInfo("bind_Error");

    SetSockAddr(&rAddr, RPort, RIP);

   do{
//-- - - - - - - - Send Mess - - - - - - - -
      puts("Send:");   gets(Msg);
      if(!strcmp(Msg,"exit")) break;
      r = sendto(S,Msg,strlen(Msg), 0,
                  (struct sockaddr far *)&rAddr, l);
      if(r == -1)  ShowInfo("Send_Error");
//-- - - - - - - - Recieve Mess - - - - - - - -
      puts("Send ok! Waiting Recieve...");
      r = recv(S, Msg, 80,0);
// r = recvfrom(SD,Msg,80,0,(struct sockaddr far *)&rAddr, &l);
                        //发送套接口的主机地址信息存放在rAddr中
      if(r == -1)  ShowInfo("Recieve_Error");
      Msg[r]=0;   puts(Msg);
      puts("Recieve ok!");

   }while(1);

   closesocket(S);
   WSACleanup();

   return ;
}
//-----------------------------------------------
```

说明:

在VC中进行WinSock API编程开发，需要使用到下面三个文件:

①winsock.h	WinSock API的头文件
②wsock32.LIB	WinSock API的连接库，把它作为项目的非缺省的连接库包含到项目文件中去。(Project -> Settings... ->Link)
③wsock32 .DLL	WinSock API的动态连接库,位于windows的系统目录下(95/98:system、NT: system32)。

```
· int sendto(              //向一指定目的地发送数据
SOCKET s,                  //源套接口
char *buf,                 //待发送数据的缓冲区
int buflen,                //缓冲区中数据的长度
int flags,                 //调用方式标志位，一般取0
struct sockaddr FAR *to,   //指向目的套接口的主机地址
int tolen                  //目的套接口主机地址的长度
```

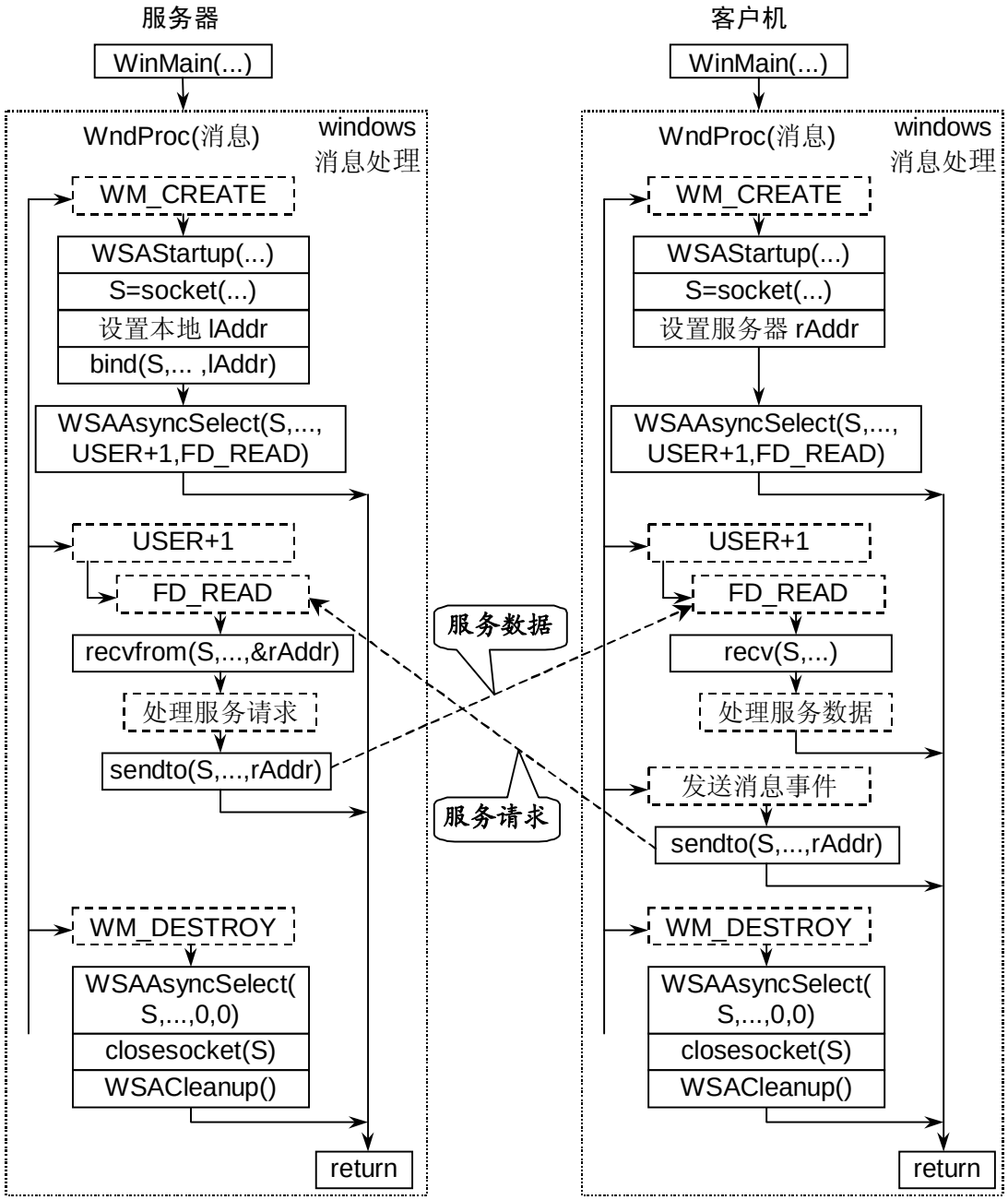
```
);
```

主要用于SOCK_DGRAM类型套接口向to参数指定端的套接口发送数据报。对于SOCK_STREAM类型套接口，to和tolen参数被忽略；这种情况下sendto()等价于send()。

```
· int recv(                    //从一个套接口接收数据
SOCKET s,                      //接收套接口
char *buf,                     //接收数据的缓冲区
int len,                       //缓冲区中数据的长度
int flags                      //调用方式标志位，一般取0
);

· int recvfrom(                //从一个套接口接收数据
SOCKET s,                      //接收套接口
char *buf,                     //接收数据的缓冲区
int len,                       //缓冲区中数据的长度
int flags                      //调用方式标志位，一般取0
struct sockaddr FAR *from,  //获取发送套接口的主机地址
int fromlen                    //发送套接口的主机地址的长度
);
```

10. 无连接协议的异步模式编程

A. 程序结构



UDP-WinSock-Windows 应用程序结构图

注:

服务器端口: 通过bind(),设置确定的服务器端口号;

服务器IP地址: 使用本地的主机IP地址;

客户机端口: 由操作系统自动分配; 可以不使用bind()。

客户机IP地址: 使用本地的主机IP地址;

服务器收到客户机的服务请求时, revcfrom(S,...,&rAddr)可获得客户机的主机地址信息rAddr, 然后, 发送sendto(S,...,rAddr)。

B. 编程示例

①服务器程序

//服务器端口：6666;
//服务器IP地址：使用本机的主机IP地址;

```
// UDPSer01 (TCP/IP) for the windows application.
/*VC6.0  File->New->Projects->选<Win32 Application>项
         (输入Project name:ServerW01->按OK按钮)->
         ->选<a sample Win32 Application>项->
         ->按Finish按钮->按OK按钮->...
//
//VC6.0 add WSOCK32.LIB in Project->Settings...->Link

#include "stdafx.h"     //VC
#include <winsock.h>    //by user
//-----------------------------------------------
char Title[]="UDPSer01";         // 窗口标题
HINSTANCE hInst;                 // current instance
LRESULT CALLBACK  WndProc(HWND, UINT, WPARAM, LPARAM);
//-----------------------------------------------
int APIENTRY WinMain(HINSTANCE hI, HINSTANCE hP, LPSTR lp, int nC)
{
   MSG msg;
   HWND hWnd;
   hInst = hI;    // Store instance handle in our global variable
   WNDCLASS wc;

   memset(&wc,0,sizeof(WNDCLASS));

   wc.lpfnWndProc    = (WNDPROC)WndProc;
   wc.hInstance      = hI;
   wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
   wc.hbrBackground  = (HBRUSH)COLOR_WINDOW;
   wc.lpszClassName  = "W1";
   RegisterClass(&wc);
//特殊窗口1，始终在顶层，任务栏显示。
   hWnd=CreateWindowEx(WS_EX_TOPMOST,"W1",Title,
      WS_DLGFRAME|WS_SYSMENU,
      200,1,200,20,
      NULL, NULL, hI, NULL);
   if (!hWnd)  return FALSE;

   ShowWindow(hWnd, nC);

   while(GetMessage(&msg, NULL, 0, 0)) // Main message loop
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
```

```
   }
   return msg.wParam;
}
//-----------------------------------------------
WSADATA ws;
SOCKET S;                          //定义套接口变量
struct sockaddr_in lAddr,rAddr;  //本地主机地址和远程主机地址变量
int LPort=6666;                    //本地端口(即服务器端口)
char LIP[]="0.0.0.0";   //IP 地址取本机的主机 IP 地址(若多个，都有效)
char Msg[88];
int d,l=sizeof(rAddr);
//-----------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family = AF_INET;            //TCP/IP协议
   A->sin_port = htons(Port);          //端口。
   A->sin_addr.s_addr =inet_addr(IP);  //IP网址。
}
//-----------------------------------------------
//消息处理
LRESULT CALLBACK WndProc(HWND hW, UINT msg,
                         WPARAM wP, LPARAM lP)
{
   switch (msg)
   {
   case WM_DESTROY:
      WSAAsyncSelect(S, hW, 0, 0);  //注销网络异步选择事件消息
      closesocket(S);               //关闭套接口
      WSACleanup();                 //卸载网络动态链接库
      PostQuitMessage(0);           //向窗口发送程序退出消息
      break;
   case WM_CREATE:
      WSAStartup(0x0101,&ws);       //装载网络动态链接库
      S=socket(AF_INET, SOCK_DGRAM,0); //创建套接口
      SetSockAddr(&lAddr,LPort,LIP);   //设置本地主机地址参数
      bind(S,(struct sockaddr *)&lAddr,sizeof(lAddr));
                                    //捆绑本地主机地址
      WSAAsyncSelect(S,hW,WM_USER+1,FD_READ);
                                    //注册网络异步选择事件消息
      break;
   case WM_USER+1:
      switch(LOWORD(lP))
      {
      case FD_READ:
         d=recvfrom(S, Msg,sizeof(Msg), 0, //接收客户机信息
                  (struct sockaddr *)&rAddr, &l);
                        // Msg=接收到的信息，d=接收到的字符数，
```

```
                        // rAddr=客户机的主机地址
          Msg[d]=0;
          char buf[88];
          wsprintf(buf,"from Client:%s",Msg);
          SetWindowText(hW,buf);     //在窗口标题栏显示接收的信息
          //把接收的信息发回给客户机
          sendto(S, Msg, strlen(Msg), 0,
                  (struct sockaddr *) &rAddr, l);
          break;
       }
       break;
   }
   return DefWindowProc(hW, msg, wP, lP);
}
//-------------------------------------------------
```

②客户机程序

```
//客户机端口: 由操作系统自动分配;
//客户机IP地址: 使用本机的主机IP地址;

// UDPCli01 (TCP/IP) for the windows application.
//
//VC6.0 add WSOCK32.LIB in Project->Settings...->Link

#include "stdafx.h"     //VC
#include <winsock.h>    //by user
//-------------------------------------------------
char Title[]="UDPCli01";         // 窗口标题
HINSTANCE hInst;                 // current instance
LRESULT CALLBACK  WndProc(HWND, UINT, WPARAM, LPARAM);
//-------------------------------------------------
int APIENTRY WinMain(HINSTANCE hI, HINSTANCE hP, LPSTR lp, int nC)
{
   MSG msg;
   HWND hWnd;
   hInst = hI;          // Store instance handle in our global
variable
   WNDCLASS wc;

   memset(&wc,0,sizeof(WNDCLASS));

   wc.lpfnWndProc    = (WNDPROC)WndProc;
   wc.hInstance      = hI;
   wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
   wc.hbrBackground  = (HBRUSH)COLOR_WINDOW;
   wc.lpszClassName  = "W1";
   RegisterClass(&wc);
//特殊窗口1，始终在顶层，任务栏显示。
   hWnd=CreateWindowEx(WS_EX_TOPMOST,"W1",Title,
      WS_DLGFRAME|WS_SYSMENU,
```

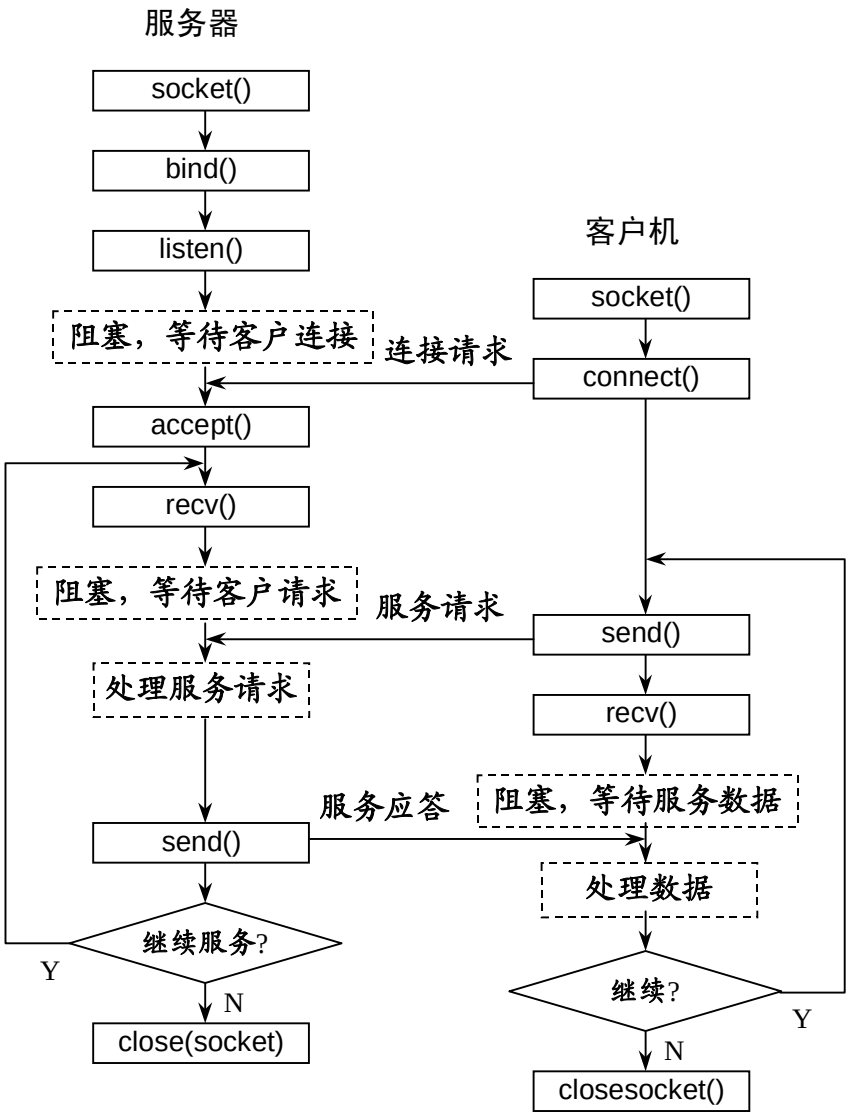
```
      400,1,200,40,
      NULL, NULL, hI, NULL);
   if (!hWnd)  return FALSE;

   ShowWindow(hWnd, nC);

   while(GetMessage(&msg, NULL, 0, 0))// Main message loop
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
//-------------------------------------------------
WSADATA ws;
SOCKET S;                       //定义套接口变量
struct sockaddr_in  rAddr;      //远程主机地址变量
int RPort=6666;                 //远程服务器端口
char RIP[16]="127.0.0.1";       //远程服务器 IP 地址
char Msg[88];
int d,l=sizeof(rAddr);
//-------------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family = AF_INET;             //TCP/IP协议
   A->sin_port = htons(Port);           //端口。
   A->sin_addr.s_addr =inet_addr(IP);  //IP网址。
}
//-------------------------------------------------
//消息处理
LRESULT CALLBACK WndProc(HWND hW, UINT msg,
                         WPARAM wP, LPARAM lP)
{
   switch (msg)
   {
   case WM_DESTROY:
      WSAAsyncSelect(S, hW, 0, 0);  //注销网络异步选择事件消息
      closesocket(S);               //关闭套接口
      WSACleanup();                 //卸载网络动态链接库
      PostQuitMessage(0);           //向窗口发送程序退出消息
      break;
   case WM_CREATE:
      WSAStartup(0x0101,&ws);       //装载网络动态链接库
      S=socket(AF_INET, SOCK_DGRAM,0); //创建套接口
      SetSockAddr(&rAddr,RPort,RIP);   //设置远程服务器主机地址参数
      WSAAsyncSelect(S,hW,WM_USER+1,FD_READ);
                                    //注册网络异步选择事件消息
      break;
```

```
   case WM_USER+1:
      switch(LOWORD(lP))
      {
      case FD_READ:
         d=recv(S,Msg,sizeof(Msg),0);  //接收服务器数据
         Msg[d]=0;
         char buf[88];
         wsprintf(buf,"from Server:%s",Msg);
         SetWindowText(hW,buf);     //在窗口标题栏显示接收的信息
         break;
      }
      break;
   case WM_LBUTTONDOWN:              //鼠标左键按下
      wsprintf(Msg,"Hello!");        //发送"Hello!"
      sendto(S,Msg,strlen(Msg),0,(struct sockaddr *)&rAddr,l);
      break;
   case WM_RBUTTONDOWN:              //鼠标右键按下
      wsprintf(Msg,"Hi!");           //发送"Hi!"
      sendto(S,Msg,strlen(Msg),0,(struct sockaddr *)&rAddr,l);
      break;
  }
   return DefWindowProc(hW, msg, wP, lP);
}
//-----------------------------------------------
```

11. 面向连接协议的同步模式编程

工作过程：

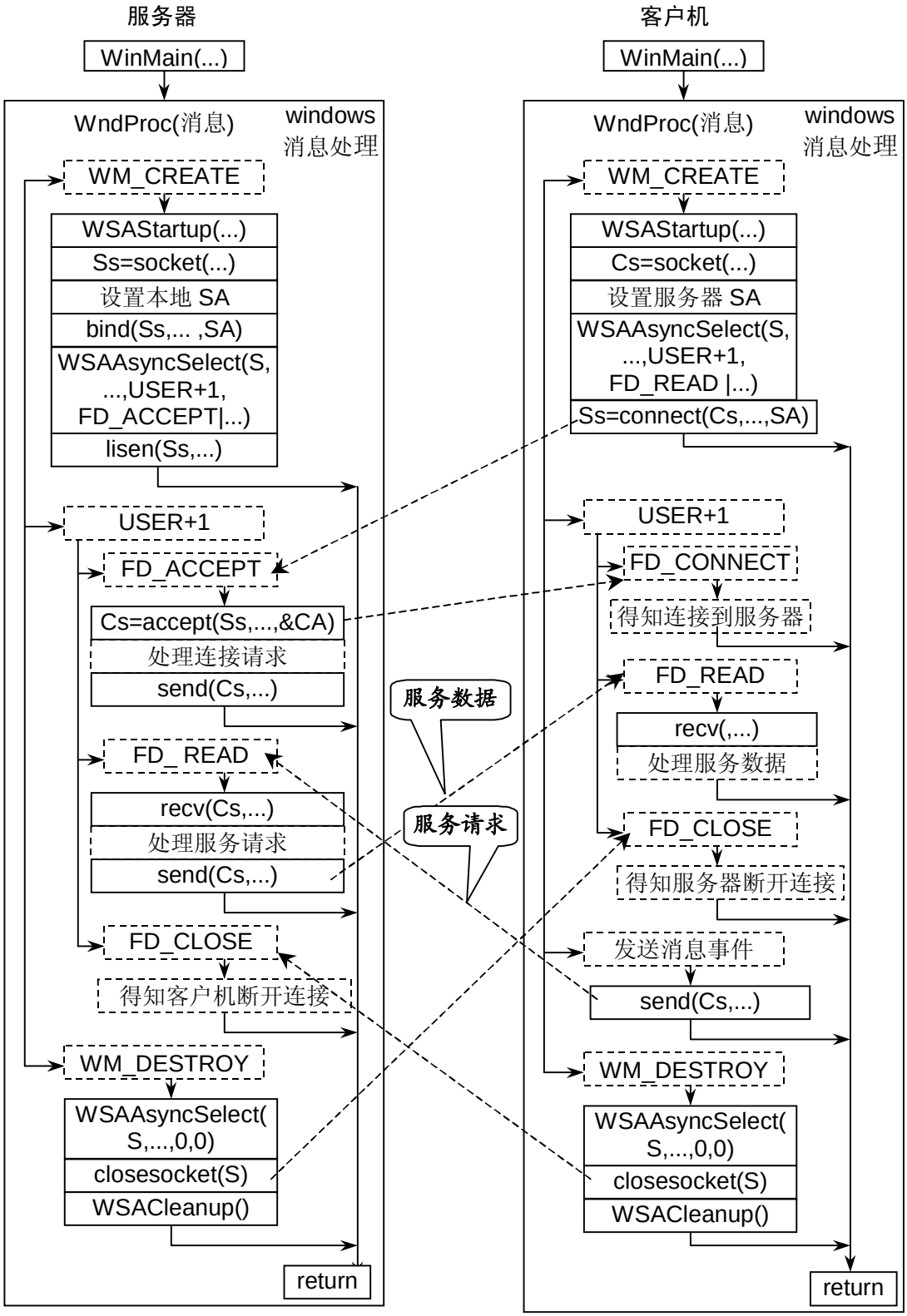
服务器首先启动,通过调用**socket()**建立一个套接口，
然后**bind()**将该套接口和本地地址(**IP**地址和端口)联系在一起，
再**listen()**使套接口做好侦听的准备，并规定它的请求队列的长度，
之后就调用**accept()**来接收连接，并获得客户机的地址信息；
客户在建立套接口后就可调用**connect()**和服务器建立连接；
连接一旦建立，客户机和服务器之间就可以通过调用：
send()和**recv()** (或**read()**和**write()**)来发送和接收数据；
最后，待数据传送结束后，双方调用**closesocket()**关闭套接口。



面向连接套接口应用程序时序图

12. 面向连接协议的异步模式编程

A. 程序结构



TCP-WinSock-Windows 应用程序结构图

2. 编程示例：

①服务器程序

```
// TCP-Server (TCP/IP) for the windows application.
/*VC6.0  File->New->Projects->选<Win32 Application>项
         (输入Project name:ServerW01->按OK按钮)->
         ->选<a sample Win32 Application>项->
         ->按Finish按钮->按OK按钮->...
*/
//VC6.0 add WSOCK32.LIB in Project->Settings...->Link

#include "stdafx.h"          //VC
#include <winsock.h>         //by user
//-----------------------------------------------
char Title[]=" TCPServer ";      // 窗口标题
HINSTANCE hInst;                 // current instance
LRESULT CALLBACK  WndProc(HWND, UINT, WPARAM, LPARAM);
//-----------------------------------------------
int APIENTRY WinMain(HINSTANCE hI,HINSTANCE hP,LPSTR lp,int nC)
{
   MSG msg;
   HWND hWnd;
   hInst = hI;   //Store instance handle in our global variable
   WNDCLASS wc;

   memset(&wc,0,sizeof(WNDCLASS));

   wc.lpfnWndProc    = (WNDPROC)WndProc;
   wc.hInstance      = hI;
   wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
   wc.hbrBackground  = (HBRUSH)COLOR_WINDOW;
   wc.lpszClassName  = "W1";
   RegisterClass(&wc);
//特殊窗口1，始终在顶层，任务栏显示。
   hWnd=CreateWindowEx(WS_EX_TOPMOST,"W1",Title,
      WS_DLGFRAME|WS_SYSMENU,
      400,1,200,40,
      NULL, NULL, hI, NULL);
   if(!hWnd)  return FALSE;

   ShowWindow(hWnd, nC);
   // Main message loop:
   while (GetMessage(&msg, NULL, 0, 0))
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
//-----------------------------------------------
```

```
WSADATA ws;
SOCKET Ss,Cs;                    //服务器和客户机的套接口变量
struct sockaddr_in SA,CA;        //服务器和客户机的主机地址变量
WORD SPort = 6666;               //本机端口(服务器)
char SIP[16]="127.0.0.1";        //本机IP地址(服务器)
char Msg[88];
int d,l=sizeof(CA);
//-------------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family = AF_INET;            //TCP/IP协议
   A->sin_port = htons(Port);          //端口。
   A->sin_addr.s_addr =inet_addr(IP); //IP网址。
}
//-------------------------------------------------
//消息处理
LRESULT CALLBACK WndProc(HWND hW,UINT msg,
                         WPARAM wP,LPARAM lP)
{
   switch (msg)
   {
   case WM_CREATE:                          //消息: 产生窗口
      WSAStartup(0x0101,&ws);
      Ss=socket(AF_INET, SOCK_STREAM,0); //创建套接口(流式)
      SetSockAddr(&SA,SPort,SIP);         //设置服务器主机地址
      bind(Ss,(struct sockaddr *)&SA,sizeof(SA));//捆绑主机地址
//向windows注册套接口Ss所产生的网络消息事件。
      WSAAsyncSelect(Ss,hW,WM_USER+1,
         FD_ACCEPT|FD_READ|FD_CLOSE);
      listen(Ss,5);                  //监听客户机连接请求
      break;
   case WM_DESTROY:                  //消息: 关闭窗口
      WSAAsyncSelect(Ss, hW, 0, 0); //注销套接口Ss的消息事件。
      closesocket(Ss);               //关闭套接口Ss
      WSACleanup();                  //卸载WinSock DLL
      PostQuitMessage(0);            //向windows发送退出程序的消息
      break;
   case WM_USER+1:
      switch(LOWORD(lP))
      {
      case FD_ACCEPT:
         Cs=accept(Ss,(struct sockaddr *)&CA,&l);
                                          //获取客户机主机地址
         wsprintf(Msg,"S:Welcome!");
         send(Cs, Msg,strlen(Msg),0);     //向客户机发送连接应答
         break;
      case FD_READ:
```

```
            d=recv(wP,Msg,sizeof(Msg),0); //接收客户机服务请求，wP=Cs
            Msg[d]=0;
            SetWindowText(hW, Msg);       //在窗口标题栏显示服务请求
            send(wP, Msg,strlen(Msg),0);  //向客户机发送服务数据
            break;
         case FD_CLOSE:
            wsprintf(Msg,"Client leave! [%d]",wP);
            SetWindowText(hW, Msg);       //在窗口标题栏显示信息
            break;
         }
         break;
   }
   return DefWindowProc(hW,msg,wP,lP);
}
//-------------------------------------------------
```

②客户机程序

```
// TCP-Client (TCP/IP) for the windows application.
//
//VC6.0 add WSOCK32.LIB in Project->Settings...->Link

#include "stdafx.h"
#include <winsock.h>          //by user
//-------------------------------------------------
char Title[]="TCPClient";        // 窗口标题
HINSTANCE hInst;                 // current instance
LRESULT CALLBACK  WndProc(HWND, UINT, WPARAM, LPARAM);
//-------------------------------------------------
int APIENTRY WinMain(HINSTANCE hI,HINSTANCE hP,LPSTR lp,int nC)
{
   MSG msg;
   HWND hWnd;
   hInst = hI; //Store instance handle in our global variable
   WNDCLASS wc;

   memset(&wc,0,sizeof(WNDCLASS));

   wc.lpfnWndProc    = (WNDPROC)WndProc;
   wc.hInstance      = hI;
   wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
   wc.hbrBackground  = (HBRUSH)COLOR_WINDOW;
   wc.lpszClassName  = "W1";
   RegisterClass(&wc);
//特殊窗口1，始终在顶层，任务栏显示。
   hWnd=CreateWindowEx(WS_EX_TOPMOST,"W1",Title,
      WS_DLGFRAME|WS_SYSMENU,
      400,1,200,40,
      NULL, NULL, hI, NULL);
   if(!hWnd)   return FALSE;
```

```
   ShowWindow(hWnd, nC);
   // Main message loop:
   while (GetMessage(&msg, NULL, 0, 0))
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
//-----------------------------------------------
WSADATA ws;
SOCKET Ss,Cs;                  //服务器和客户机的套接口变量
struct sockaddr_in SA,CA;      //服务器和客户机的主机地址变量
WORD SPort = 6666;             //远程服务器端口
char SIP[16]="127.0.0.1";      //远程服务器IP地址
char Msg[88];
int d,l=sizeof(CA);
//-----------------------------------------------
void SetSockAddr(struct sockaddr_in *A,WORD Port,char *IP)
{
   A->sin_family = AF_INET;            //TCP/IP协议
   A->sin_port = htons(Port);          //端口。
   A->sin_addr.s_addr =inet_addr(IP);  //IP网址。
}
//-----------------------------------------------
//消息处理
LRESULT CALLBACK WndProc(HWND hW,UINT msg,
                         WPARAM wP,LPARAM lP)
{
   switch (msg)
   {
   case WM_CREATE:
      WSAStartup(0x0101,&ws);
      Cs=socket(AF_INET,SOCK_STREAM,0);   //创建套接口(流式)
      SetSockAddr(&SA,SPort,SIP);         //服务器主机地址(远程)
      WSAAsyncSelect(Cs,hW,WM_USER+1,
                     FD_CONNECT|FD_READ|FD_CLOSE);
      connect(Cs,(struct sockaddr *)&SA, sizeof(SA));
                                          //连接服务器
      break;
   case WM_DESTROY:                  //消息: 关闭窗口
      WSAAsyncSelect(Cs, hW, 0, 0); //注销套接口Cs的消息事件。
      closesocket(Cs);               //关闭套接口Cs
      WSACleanup();                  //卸载WinSock DLL
      PostQuitMessage(0);            //向windows发送退出程序的消息
      break;
   case WM_USER+1:
```

```
      switch(LOWORD(lP))
      {
      case FD_CONNECT:
         SetWindowText(hW,"已连接到服务器！");//在窗口标题栏显示信息
         break;
      case FD_READ:
         d=recv(wP,Msg,sizeof(Msg),0); //接收服务器的服务数据，
                                       //wP=Cs
         Msg[d]=0;
         SetWindowText(hW, Msg);    //在窗口标题栏显示服务数据
         break;
      case FD_CLOSE:
         wsprintf(Msg,"Server Stop! [%d]",wP);
         SetWindowText(hW, Msg);    //在窗口标题栏显示信息
         break;
      }
      break;
   case WM_LBUTTONDOWN:             //消息: 鼠标左键按下
      wsprintf(Msg,"Hello!");
      send(Cs, Msg,strlen(Msg),0);  //向服务器发送服务请求"Hello!"
      break;
   case WM_RBUTTONDOWN:             //消息: 鼠标右键按下
      wsprintf(Msg,"Hi!");
      send(Cs, Msg,strlen(Msg),0);  //向服务器发送服务请求"Hi!"
      break;
   }
   return DefWindowProc(hW,msg,wP,lP);
}
//-----------------------------------------------
```