西北師範大學
NORTHWEST NORMAL UNIVERSITY

计算机网络

运输层编程（5）

陈旺虎
chenwh@nwnu.edu.cn

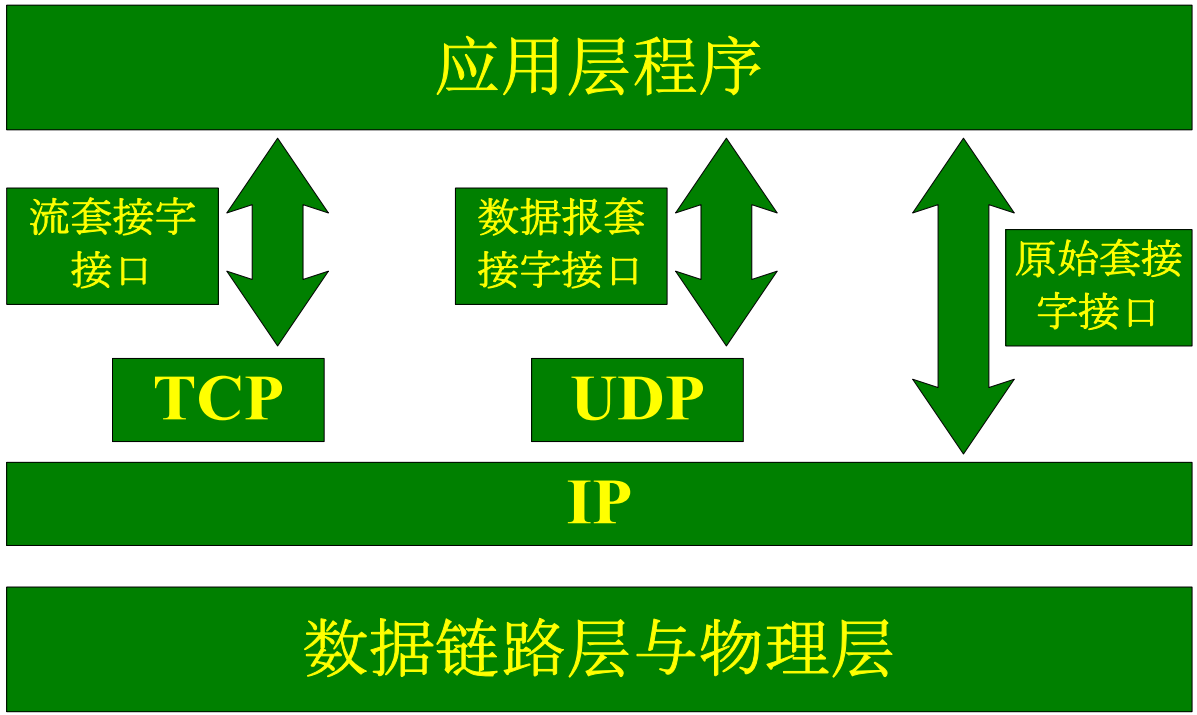
Review

- TCP协议格式
- TCP可靠传输

为什么需要三次握手？

- A发送一次确认的原因
  - 应对出现“已失效的连接请求报文段”的情况，即防止已失效的连接请求报文段突然又传到了B。
- 例1：A发出连接请求，但该请求丢失，A重传连接请求，到达B，则正常；

一. 认识Socket



Socket类型

- 流式套接字(SOCK_STREAM)
  - 面向连接、可靠的数据传输服务
  - 内设置流量控制，数据被看作是字节流，无长度限制
- 数据报套接字(SOCK_DGRAM)
  - 无连接服务
  - 数据包以独立数据报的形式被发送，无差错保证，数据可能丢失或重复，可能失序
- 原始套接字(SOCK_RAW)
  - 可以对较低层次协议，如IP、ICMP直接访问

两类系统中使用的Socket

- 不同操作系统中的Socket
  - Windows Socket (Winsock)
  - Linux Socket (BSD Socket)

Windows Socket

- 简称Winsock
- Windows环境下使用的一套网络编程规范
- 基于4.3 BSD的BSD Socket API制定
  - 1991年Winsock 1.1，16位
  - 1997年Winsock 2.2 版，32位，由WSOCK32.DLL支持，主要用在Windows 98及以后的版本中
  - 已经成为Windows环境下网络编程的事实标准

Socket常用函数

- 基本函数 （与BSD Socket兼容）
- 网络信息检索函数

基本函数

- 网络连接函数
  - socket　创建套接字
  - bind　绑定本机端口
  - Connect　建立连接
  - listen　监听端口
  - accept　接受连接
  - recv，recvfrom　数据接收
  - send，sendto　数据发送
  - close，shutdown　关闭套接字

基本函数

- IP地址转换函数
  - inet_addr()
    - 点分十进制数表示的IP地址转换为网络字节序的IP地址
  - inet_ntoa()
    - 网络字节序的IP地址转换为点分十进制数表示的IP地址
- 字节排序函数
  - htonl　4字节主机字节序转换为网络字节序
  - Ntohl　4字节网络字节序转换为主机字节序
  - htons　2字节主机字节序转换为网络字节序
  - ntohs　2字节网络字节序转换为主机字节序

网络信息检索函数

- gethostname　获得主机名
- getpeername　获得与套接口相连的远程协议地址
- getsockname　获得套接口本地协议地址
- gethostbyname　根据主机名取得主机信息
- gethostbyaddr　根据主机地址取得主机信息
- getprotobyname　根据协议名取得主机协议信息
- getprotobynumber　根据协议号取得主机协议信息
- getservbyname　根据服务名取得相关服务信息
- getservbyport　根据端口号取得相关服务信息
- getsockopt/setsockopt　获取/设置一个套接口选项
- ioctlsocket　设置套接口的工作方式

Windows中的Socket编程

- Winsock 的启动
- Winsock API基本函数
- TCP/IP网络程序框架(C/S模式)
- 阻塞与非阻塞通信方式
- 实例程序说明

Winsock

- 需要包含
  - 头文件Winsock2.h
  - 库ws2_32.lib
  - 包含办法可以用语句来告诉编译时调用该库

    #pragma comment(lib,"ws2_32.lib");

- 集成开发环境（如Visual C++ 6.0）
  - “工程”>“设置”>“工程设置”>“链接”>“对象/库模块”中加入“ws2_32.lib”

Windows Socket的启动

- 检查协议栈安装情况

```
int WSAStartup(
        WORD      wVersionRequested,
        LPWSADATA lpWSAData
);
```

wVersionRequested，双字节型数值，版本号，对Winsock2.2而言，其值为0x0202，也可以用宏MAKEWORD(2,2)来获得；

lpWSAData，指向WSADATA结构的指针，返回Winsock实现的详细信息

Winsock启动示例

```
#include <Winsock2.h>
WORD wVersionRequested;
WSADATA wsaData;
wVersionRequested=MAKEWORD(2,2);
if(WSAStartup(wVersionRequested, &wsaData) !=0 )
        //Winsock初始化错误
        return;
if(wsaData.wVersion != wVersionRequested) {
        //Winsock版本不匹配
        WSACleanup();
        return;
} //说明WinsockDLL正确加载，可以执行以下代码
```

创建套接口socket()

SOCKET socket(int af, int type, int protocol);

- af ：协议地址族，对于TCP或UDP，用常量AF_INET表示使IP地址
- type：描述套接口的类型，SOCK_STREAM、SOCK_DGRAM、SOCK_RAW
- Protocol：af和type确定后，协议字段的值是限定的

协议	地址族	套接口类型	套接口类型使用的值	协议字段
互联网协议(IP)	AF_INET	TCP	SOCK_STREAM	IPPROTO_TCP
		UDP	SOCK_DGRAM	IPPROTO_UDP
		Raw	SOCK_RAW	IPPROTO_RAW IPPROTO_ICMP

绑定本地地址—bind()

- 绑定socket与该主机上提供服务的某端口

```
int bind(SOCKET s, const struct sockaddr FAR * name, int namelen);
```

- S：套接口描述字，socket()返回的值口号

Name：地址结构指针，存储了套接口的地址信息

```
struct sockaddr_in {
    short  sin_family;  //一般为AF_INET，表示IP地址族
    u_short sin_port;     //网络字节序16位端口号
    struct in_addr  sin_addr;//网络字节序32位IP地址
    char          sin_zero[8];   //用0填充
}
```

namelen表示name的长度

- port为0，则由系统自动指派一个1024~5000之间惟一的端口号

bind()实例

```
SOCKET s;
sockaddr_in tcpaddr;
int iSockErr;
int port=5000; //端口号
s=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
tcpaddr.sin_family=AF_INET;
tcpaddr.sin_port=htons(port);
tcpaddr.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(s, (LPSOCKADDR)&tcpaddr,sizeof(tcpaddr)) ==
    SOCKET_ERROR){
    iSockErr=WSAGetLastError();
    ……
    return;
}
```

服务器端启动监听—listen()函数

```
int listen(
        SOCKET s,
        int backlog
);
```

s：已绑定地址，还未建立连接的套接口描述字

backlog：指定了正在等待连接的最大队列长度

客户端请求连接—connect()函数

- 客户端调用connect()函数来与服务器建立连接

```
int connect(
        SOCKET s,
        const struct sockaddr FAR * name,
        int namelen
);
```

- S：将要建立连接的套接口描述字
- Name：指向远端套接口地址结构(sockaddr_in)的指针，表示s套接口欲与其建立一条连接
- Namelen：是服务器端的地址长度，即name的长度

服务器端接受连接—accept()函数

- 在服务器端调用accept()函数时，表示可以接收来自客户端发出的连接请求，双方进入连接状态

```
SOCKET accept(
        SOCKET s,     //监听套接字
        struct sockaddr FAR * addr,
        int FAR * addrlen
);
```

- saddr是一个地址结构的指针，用来存放发出连接请求的那个客户机的IP地址信息
- addrlen指出客户套接口地址结构的长度
- 函数说明：该函数用于面向连接的服务器端

发送数据-send()函数

- 在已经建立连接的套接口上发送数据

```
int send(
        SOCKET s, //标识已建立连接的套接字
        const char FAR * buf,
        int len,
        int flags);
```

- flags：数据传输方式
  - 0表示按正常方式发送数据；宏MSG_DONTROUTE说明目标主机就在本地网络中，无需路由选择；MSG_OOB指出数据是按带外数据发送的(置URG字段)
- 函数说明：对于数据报类型套接口必须注意发送数据长度不大于通信子网的IP包最大长度

接收数据—recv()函数

- 已建立连接的套接口上接收数据

```
int recv(
        SOCKET s,
        char FAR * buf,
        int len,
        int flags
);
```

- flags
  - 0表示接收的是正常数据，无特殊行为
  - MSG_PEEK表示没有从系统缓冲区中将数据删除
  - MSG_OOB表示处理带外数据

无连接的套接口上接收数据-recvfrom()

```
int recvfrom(
        SOCKET s,
        char FAR * buf,
        int len,
        int flags,
        struct sockaddr FAR * from,
        int FAR * fromlen
);
```

在无连接套接口上发送数据-sendto()

- 对于无连接的套接口来说，要从套接口上发送一个数据报，就要使用sendto()函数

```
int sendto(
        SOCKET s,
        const char FAR * buf,
        int len,
        int flags,
        const struct sockaddr FAR * to,
        int tolen
);
```

关闭读写通道-shutdown()函数

- 好处是当双方不再有数据要发送或接收时，可以通知对方，以防止数据丢失，并能“优雅”地关闭连接

```
int shutdown(
        SOCKET s,
        int how
);
```

shutdown()函数参数说明

- S：标识一个套接口的描述字
- How：标志，用于描述禁止哪些操作

关闭方式	参数值	说　　明
SD_RECEIVE	0	表示不允许再调用接收函数，它关闭读通道。套接口接收缓冲区中的所有数据都被丢弃，并且有新数据到达套接口时，也被TCP协议层丢弃，但它对发送缓冲区没有影响，进程仍然可以在套接口上发送数据
SD_SEND	1	表示不允许再调用发送函数，它写通道关闭。在套接口发送缓冲区中的数据都被发送出去，得到接收端确认之后，就生成一个FIN包关闭连接。但它对接收缓冲区没有影响，进程仍然可以在套接口上接收数据
SD_BOTH	2	关闭读写通道，相当于执行了上面SD_RECEIVE和SD_SEND两个命令

关闭套接口-closesocket()函数

- 关闭这个套接口，释放资源，包括等候处理的数据。

```
int closesocket(
        SOCKET s
);
```

- 参数s表示即将被关闭的套接口

IP地址转换函数

- char * inet_ntoa (struct in_addr in)

  in为传入参数，表示一个结构型的IP主机地址，该函数将IP地址转换成点分十进制IP地址字符串

- unsigned long inet_addr(const char FAR * cp)

  该函数将一个点分十进制IP地址字符串转换成32位数字表示的IP地址

字节序转换函数

- u_long htonl(u_long hostlong)
  - 4字节主机字节序表示的整数转换为4字节相应的网络字节序表示的整数
- u_short htons(u_short hostshort)
  - 2字节主机字节序表示的整数转换为2字节相应的网络字节序表示的整数
- u_long ntohl(u_long netlong)
  - 4字节网络字节序表示的整数转换为4字节相应的主机字节序表示的整数
- u_short ntohs(u_short netshort)
  - 2字节网络字节序表示的整数转换为2字节相应的主机字节序表示的整数

终止使用Winsock——WSACleanup()函数

- 当应用程序不再使用Winsock API中的任何函数时，释放为此应用程序或DLL分配的任何资源。

```
int WSACleanup(void);
```

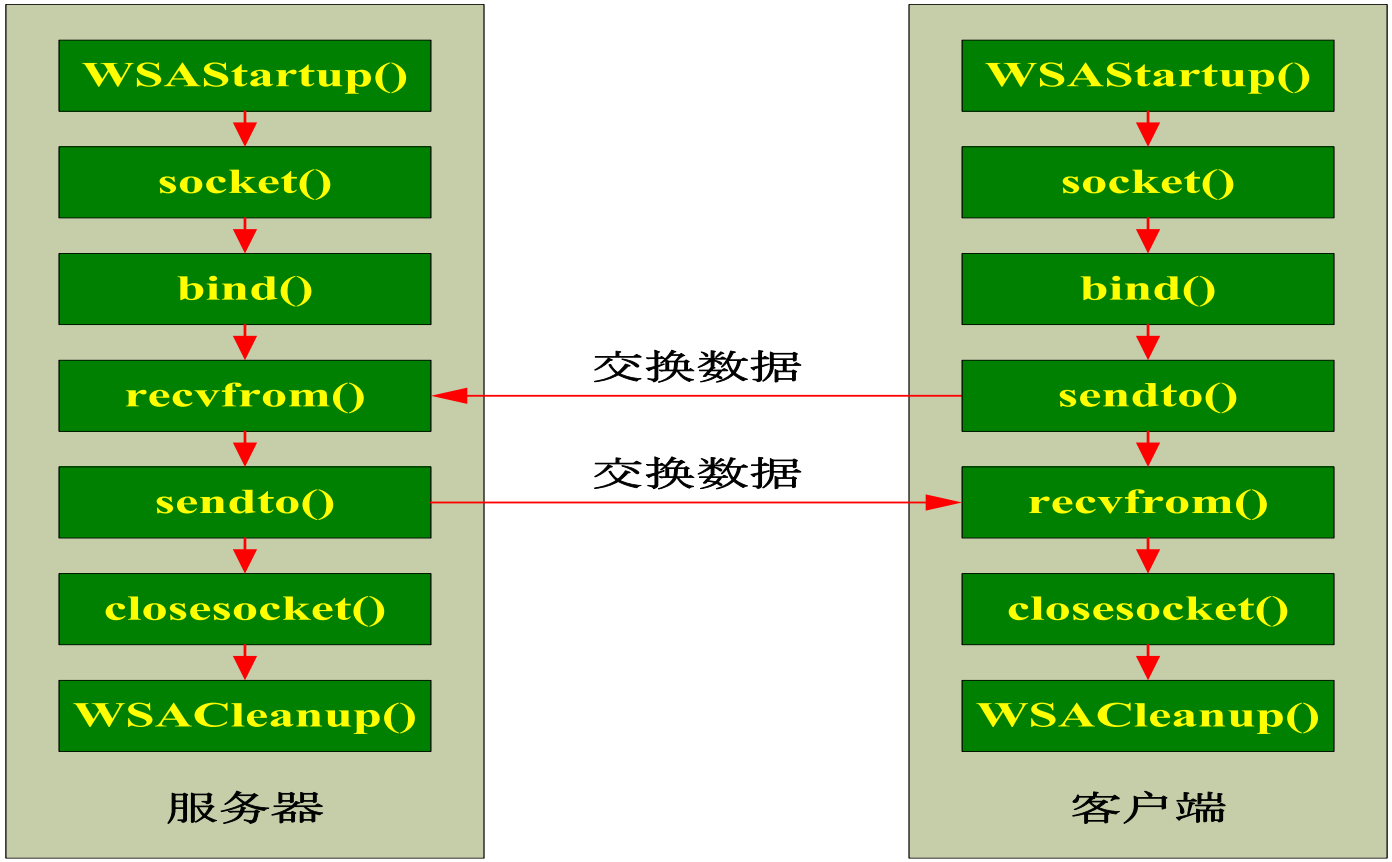
- 函数说明
  - WSACleanup()函数是任何一个Winsock应用程序在最后必须要调用的函数。
  - 在一个多线程的环境下，WSACleanup()函数中止了Windows Sockets在所有线程上的操作

TCP/IP网络程序框架

- 面向连接的C/S程序工作流程
- 无连接的C/S程序工作流程

面向连接的C/S程序工作流程图(TCP)

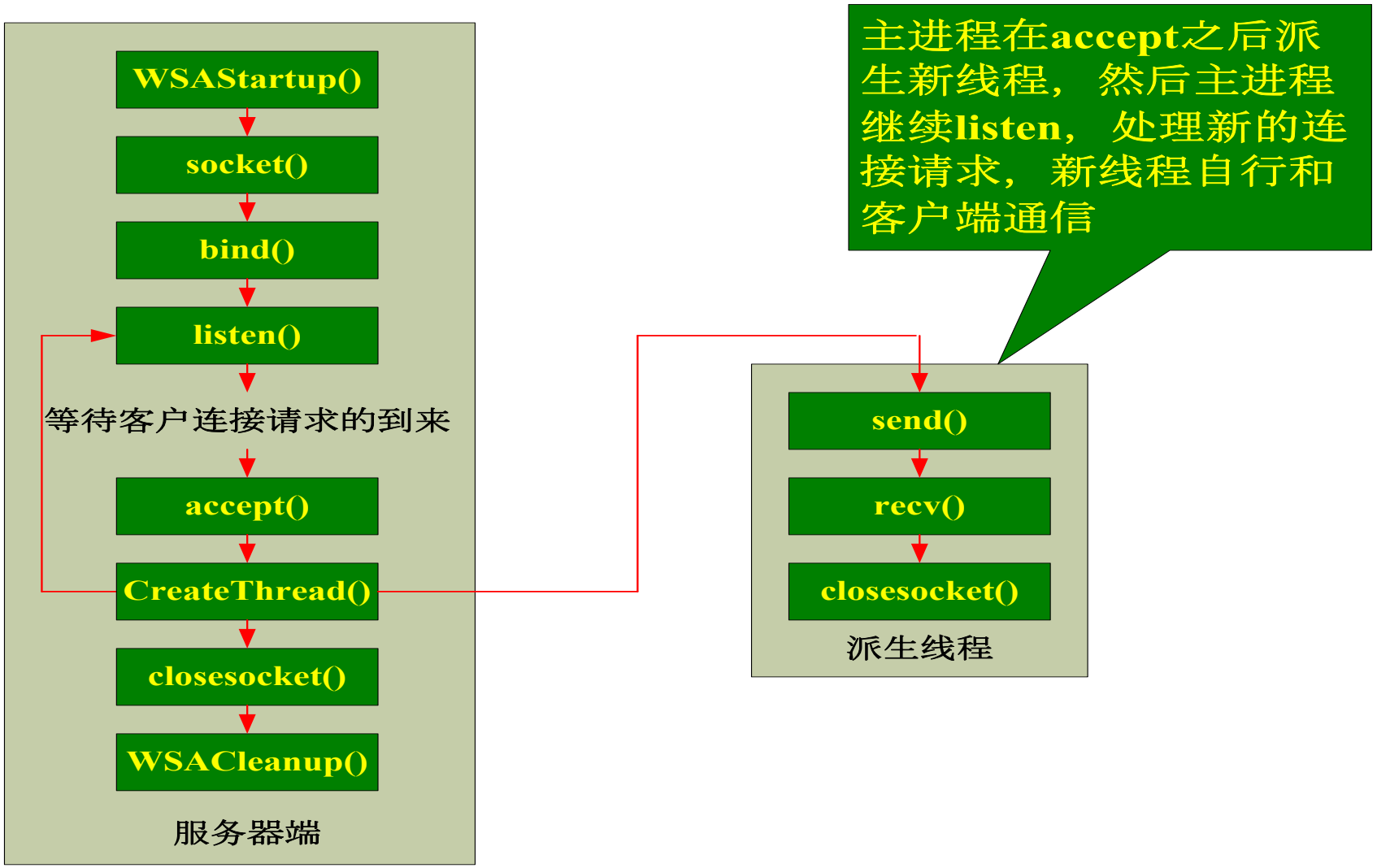
无连接的C/S程序工作流程图(UDP)



阻塞通信与非阻塞通信

- 阻塞方式
  - 套接字进行I/O操作时，函数要等待到相关的操作完成以后才能返回
  - 对提高处理机的利用率不利
  - 阻塞方式编程简单，一个套接口的默认操作模式为阻塞，可以调用函数ioctlsocket()进行设置
- 非阻塞方式
  - 套接字进行I/O操作时，无论操作成功与否，调用都会立即返回

并发服务器


主进程在accept之后派生新线程，然后主进程继续listen，处理新的连接请求，新线程自行和客户端通信
WSAStartup()
socket()
bind()
listen()
等待客户连接请求的到来
accept()
CreateThread()
closesocket()
WSACleanup()
服务器端
send()
recv()
closesocket()
派生线程

基于TCP的C/S—服务器代码

```
// server.cpp : 定义控制台应用程序的入口点。
#include "stdafx.h"
#include <Winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#define DEFAULT_PORT 5050 //服务端默认端口
int _tmain(int argc, char* argv[])
{
    int                 iPort = DEFAULT_PORT;
    WSADATA             wsaData;
    SOCKETsListen,sAccept;
    int                 iLen; //客户地址长度
    int                 iSend;//发送数据长度
    char                buf[] = "I am a server";//要发送给客户的信息
    struct sockaddr_in ser,cli;//服务器和客户的地址
    if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
    {
        printf("Failed to load Winsock.\n");
        return -1;
    }
```

```
sListen = socket(AF_INET,SOCK_STREAM,0);//创建服务器端套接口
if(sListen == INVALID_SOCKET) {
        printf("socket() Failed: %d\n",WSAGetLastError());
        return -1;
}
//以下建立服务器端地址
ser.sin_family = AF_INET; //使用IP地址族
//使用htons()把双字节主机序端口号转换为网络字节序端口号
ser.sin_port = htons(iPort);
//htonl()把一个四字节主机序IP地址转换为网络字节序主机地址
//使用系统指定的IP地址INADDR_ANY
ser.sin_addr.s_addr = htonl(INADDR_ANY);
//bind()函数进行套接定与地址的绑定
if(bind(sListen,(LPSOCKADDR)&ser,sizeof(ser)) == SOCKET_ERROR {
        printf("bind() Failed: %d\n",WSAGetLastError());
        return -1;
}
```

```
//进入监听状态
if(listen(sListen,5) == SOCKET_ERROR){
        printf("lisiten() Failed: %d\n",WSAGetLastError());
        return -1;
}
//初始化客户地址长度参数
iLen = sizeof(cli);
//进入一个无限循环，等待客户的连接请求
while(1){
        sAccept = accept(sListen,(struct sockaddr *)&cli,&iLen);
        if(sAccept == INVALID_SOCKET) {
                printf("accept() Failed: %d\n",WSAGetLastError());
                return -1;
        }
        //输出客户IP地址和端口号
        printf("Accepted client
IP:[%s],port:[%d]\n",inet_ntoa(cli.sin_addr),ntohs(cli.sin_port));
```

```
            //给连接的客户发送信息
            iSend = send(sAccept,buf,sizeof(buf),0);
            if(iSend == SOCKET_ERROR) {
                        printf("send() Failed: %d\n",WSAGetLastError());
                        break;
            }
            else if(iSend == 0) {
                        break;
            }
            else {
                        printf("send() byte: %d\n",iSend);
            }
            closesocket(sAccept);
      }
      closesocket(sListen);
      WSACleanup();
      return 0;
}
```

基于TCP的C/S—客户端代码

```
// client.cpp : 定义控制台应用程序的入口点。
#include "stdafx.h"
#include <Winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#define DATA_BUFFER 1024 //默认缓冲区大小
int _tmain(int argc, char * argv[]) {
    WSADATA wsaData;
    SOCKET sClient;
    int iPort = 5050;
    int iLen;//从服务器端接收的数据长度
    char buf[DATA_BUFFER];//接收数据的缓冲区
    struct sockaddr_in ser;//服务器端地址
    //判断参数输入是否正确： client [Server IP]
    if(argc<2) {
        //提示在命令行中输入服务器IP地址
        printf("Usage: client [server IP address]\n");
        return -1;
    }
```

```
memset(buf,0,sizeof(buf));//接收缓冲区初始化
if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
{
        printf("Failed to load Winsock.\n");
        return -1;
}
//填写要连接的服务器地址信息
ser.sin_family = AF_INET;
ser.sin_port = htons(iPort);
//inet_addr()将命令行中输入的点分IP地址转换为二进制表示的网络字节序IP地址
ser.sin_addr.s_addr = inet_addr(argv[1]);
//建立客户端流式套接口
sClient = socket(AF_INET,SOCK_STREAM,0);
if(sClient == INVALID_SOCKET)
{
        printf("socket() Failed: %d\n",WSAGetLastError());
        return -1;
```

```
//请求与服务器端建立TCP连接
if(connect(sClient,(struct sockaddr *)&ser,sizeof(ser)) == INVALID_SOCKET) {
        printf("connect() Failed: %d\n",WSAGetLastError());
        return -1;
}
else{
        //从服务器端接收数据
        iLen = recv(sClient,buf,sizeof(buf),0);
        if(iLen == 0)
                return -1;
        else if(iLen == SOCKET_ERROR){
                printf("recv() Failed: %d\n",WSAGetLastError());
                return -1;
        }
        else
                printf("recv() data from server: %s\n",buf);
}
```

```
    closesocket(sClient);
    WSACleanup();
    return 0;
}
```

高级网络编程API

- MFC编程技术定义了用于网络编程的Winsock类，类名为CAsyncSocket；还定义了一个派生于CAsyncSocket的CSocket类。
- 与前面的介绍相似，使用MFC的Winsock类进行操作时需要使用Winsock2.h、Winsock32.dll和ws2_32.lib三个文件

Thanks , Any Question?

The End.

chenwh@nwnu.edu.cn