KHỞI ĐỘNG WINSOCK  
  
Để lập trình được Winsock chúng ta sẽ khai báo thư viện winsock2.h (chứa các prototypes) và 1 file lib (chính là file .cpp đã được biên dịch thành .lib) có tên là ws2\_2.lib.

#include <stdio.h>  
#include <tchar.h>  
...  
#include <winsock2.h>  
#pragma comment (lib,"ws2\_32.lib")

**Khởi tại winsock**

int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData);  
  
Trong đó:  
- wVersionRequested là phiên bản thư viện mà mình sử dụng. Ở đây sẽ là giá trị 0x0202 có nghĩa là phiên bản 2.2. Chúng ta có thể dùng macro MAKEWORD(2,2) để trả về giá trị 0x0202.  
- lpWSData là một số thông tin bổ sung sẽ được trả về sau khi gọi khởi tạo Winsock.:  
  
  
typedef struct WSAData {   
WORD wVersion; // Phiên bản hiện tại  
WORD wHighVersion; // Phiên bản có thể hỗ trợ  
char szDescription[WSADESCRIPTION\_LEN + 1]; // Ghi chú  
char szSystemStatus[WSASYS\_STATUS\_LEN + 1]; // Trạng thái hệ thống  
  
unsigned short iMaxSockets; // Không sử dụng từ Version 2 trở đi  
unsigned short iMaxUdpDg; // Không sử dụng từ Version 2 trở đi  
char FAR \* lpVendorInfo; // Không sử dụng từ Version 2 trở đi  
} WSADATA, FAR \* LPWSADATA;

**là hàm hủy Winsock khi kết thúc chương trình.**

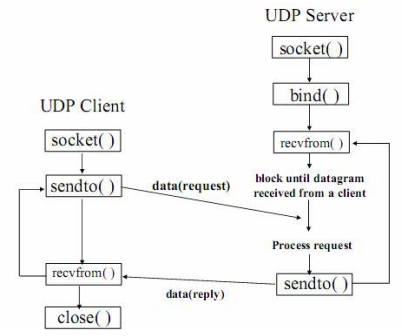
vi dụ

int WSACleanup (void);  
  
Chương trình đầu tiên:  
  
#include "stdafx.h"  
  
using namespace std;  
  
int \_tmain(int argc, \_TCHAR\* argv[])  
{  
WSADATA SData;  
int iResult = WSAStartup(0x0202,&SData);  
if (iResult!=0){  
cout << "KHONG THE KHOI DONG WINSOCK";  
return 1;  
}  
  
cout << "KHOI TAO SOCKET THANH CONG: \n";  
cout << "Phien ban: "<< SData.wVersion << "\n";   
cout << "Phien ban co the ho tro: "<< SData.wHighVersion << "\n";  
cout << "Ghi chu: " << SData.szDescription << "\n";  
cout << "Thong tin cau hinh: " << SData.szSystemStatus << "\n";  
  
WSACleanup();  
return 0;  
}

**2. Khởi tạo Socket**

Chúng ta sử dụng cấu trúc SOCKET để lưu giữ 1 Socket. Và có thể sử dụng hàm sau đây để tạo Socket.  
  
SOCKET socket (  
int af,  
int type,  
int protocol  
);  
  
Ví dụ:   
SOCKET s = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_IP);  
  
Trong đó:  
\* af: Là một con số ID để quyết định Socket của chúng ta sử dụng giao thức (protocol) để kết nối.  
- AF\_INET : TCP/IP (Phổ biến nhất hiện nay -> dùng địa chỉ IP để truyền dữ liệu)  
- AF\_NETBIOS: NetBIOS (Giao thức dùng tên máy để truyền dữ liệu)  
- AF\_APPLETALK: AppleTalk  
- AF\_ATM: ATM

**Giao thức udp\_ip**



- SOCK\_DGRAM: Hay là giao thức UDP. Khi chương trình chúng ta dùng UDP để truyền dữ liệu thì chuyện gì sẽ xảy ra giữa bên gởi và bên nhận? Bên gửi cứ gửi và gửi và nó không hề quan tâm tới vấn đề bên nhận có nhận được nó hay không?  
=> Ưu điểm: Tốc độ truyền dữ liệu nhanh.  
=> Nhược điểm: Khả năng sai, mất dữ liệu sẽ rất lớn.  
  
dùng UDP khi

- Những ứng dụng cần dữ liệu tức thời như:  
- Chương trình nghe nhạc trực tuyến. Vấn đề sai bit (vấp khi nghe nhạc) không quan trọng mấy vì yêu cầu của nó là đảm bảo tốc độ nhanh.  
- Chương trình Chat chẳn hạn.

- SOCK\_STREAM: Đây là giao thức TCP. Nó ngược với UDP vì nó đảm bảo giữa bên gửi và bên nhận dữ liệu phải chính xác. Vì vậy 2 bên sẽ phải bắt tay rất nhiều lần khi truyền được dữ liệu (ví dụ như bên gửi sẽ gửi n gói tin (packet), bên nhận sẽ kiểm tra có bị mất hay sai gói tin nào hay không, nếu đủ thì nó sẽ yêu cầu bên gửi gửi tiếp n gói tin tiếp theo, ngược lại thì nó sẽ yêu cầu gửi lại)  
=> Ưu điểm: Chất lượng gởi tin cậy.  
=> Nhược điểm: Chậm hơn UDP.  
Những ứng dụng như WEB, MAIL, FTP,…

- SOCK\_RAW:   
Là giao thức để kiểm soát mạng, kiểm tra kết nối…

1 so ham lay thông tin của socket

3. Một số hàm lấy thông tin về mạng  
  
a. Lấy thông tin Socket   
  
  
int WSAEnumProtocols (  
LPINT lpiProtocols,   
LPWSAPROTOCOL\_INFO lpProtocolBuffer,   
LPDWORD lpdwBufferLength  
);  
  
  
lpiProtocols: NULL  
lpProtocolBuffer: Kiểu dữ liệu trả về  
lpdwBufferLength: Kích thước của kiểu dữ liệu

b. Lấy tên máy tính của mình  
  
  
int gethostname(char\* name, int namelen);

địa chỉ ip

Địa chỉ IP là 1 con số 4 byte để xác định 1 host trên mạng.

Sử dụng ip để láy tên máy

Sử dụng : inet\_addr và inet\_ntoa để chuyển đổi qua lại giữa u\_long và char\*  
  
  
u\_long YahooAddr = inet\_addr("216.109.112.135");  
cout << "IP: " << inet\_ntoa(\*(in\_addr\*) &YahooAddr) << "\n";

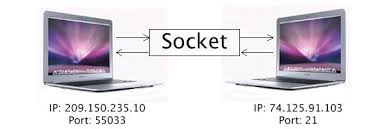
**+ , Lấy IP theo tên máy**  
  
  
  
struct hostent\* FAR gethostbyname(const char\* name);  
  
Trong đó  
  
typedef struct hostent {  
char FAR\* h\_name; // Tên máy tính  
char FAR FAR\*\* h\_aliases; // Bí danh máy tính  
short h\_addrtype; // Kiểu IP (AF\_INET)  
short h\_length; // Kích thước IP  
char FAR FAR\*\* h\_addr\_list; // Danh sách các địa chỉ IP   
// 1 host có thể có 1 hoặc nhiều IP  
} HOSTENT,  
  
  
Ví dụ như:   
  
char lpHostName[100];  
hostent \*MyPC;   
gethostname(lpHostName,100);  
MyPC = gethostbyname(lpHostName);

Nguyên nhân dùng ip

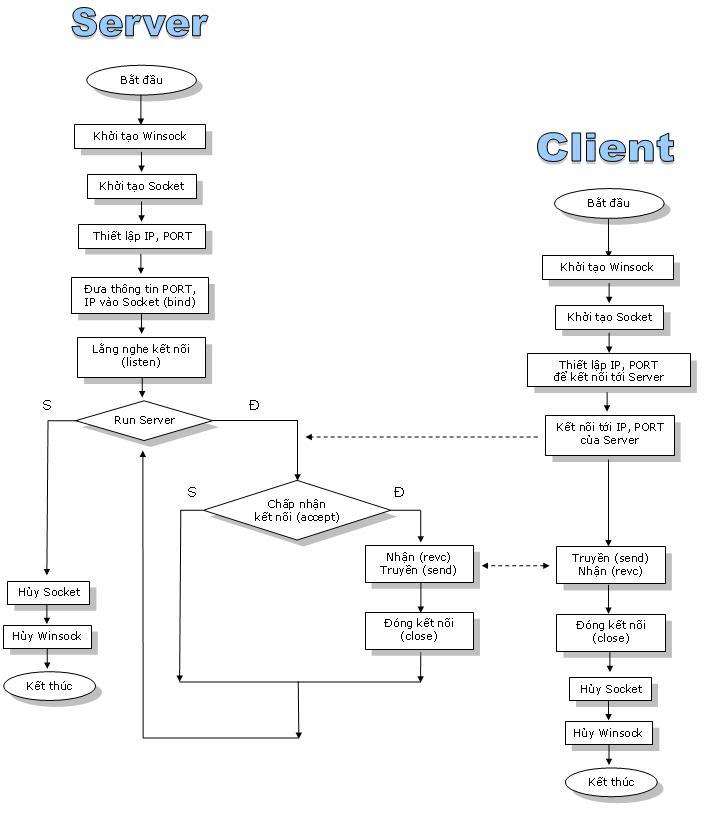
- Tại sao lại phân loại địa chỉ IP  
 - Là do nhu cầu của từng môi trường mạng.

+ Tại sao là 8 bit nhưng lại là 2^7 ? -> Là vì số bit từ làm Subnet là 0 -> 7 (biểu diễn tới 2^7)

**Socket**



**Nguyên lý hoạt đọng của sócket**



 BIND  
  
  
int bind(  
SOCKET s,   
const struct sockaddr FAR\* name,   
int namelen  
);  
  
  
Tác dụng dụng của BIND là sẽ giúp cho SOCKET của SERVER biết rằng nó sẽ chờ đợi kết nối và nhận dữ liệu trên IP nào và PORT bao nhiêu?  
  
  
PORT nên chọn ở đây nên ở trong khoảng nào:  
- 0 -1023: Là những PORT đã được sử dụng bởi các dịch vụ như WEB, FTP, …  
- 1024-49151: Là PORT dành cho SERVER lắng nghe. SERVER nên chọn trong khoảng này.  
- 49152-65535: Là PORT khởi tạo ngẫu nhiên dành cho CLIENT kết nối tới Server.

**Các hàm**   
  
Hàm Bind gồm có 3 thông số:   
- SOCKET s: Socket được thiết lập   
- sockaddr name: Cấu trúc ADDR bao gồm địa chỉ IP và PORT  
- int namelen: Kích thước của cấu trúc sockaddr

**LISTEN**  
  
int listen(  
SOCKET s,   
int backlog  
);  
  
  
Kể từ khi gọi hàm này thì SERVER sẽ bắt đầu lắng nghe kết nối của mình.  
  
Hàm LISTEN gồm có 2 thông số:   
- SOCKET s: Socket đã được thiết lập IP và PORT.  
- int backlog: Số kết nối cho phép chờ trong hàng đợi khi Server chưa chấp nhận kết nối. (vì đôi lúc có thể có tới 2 hay 3 client kết nối tới cùng 1 lúc). Giá trị tốt nhất là khoảng từ 5 – 10.

**c. CONNECT**   
  
  
int connect(  
SOCKET s,   
struct sockaddr \*serv\_addr,   
int addrlen );  
  
  
  
Hàm được gọi từ CLIENT nếu nó muốn kết nối tới SERVER  
- SOCKET s: Socket đã được khởi tạo.  
- sockaddr \*serv\_addr: IP và PORT của Server.  
- int addrlen: Sizeof của cấu trúc sockaddr.

**d. ACCEPT**  
  
  
  
SOCKET accept(  
SOCKET s,   
struct sockaddr FAR\* addr,   
int FAR\* addrlen  
);  
  
  
Khi Client kết nối tới Server. Nó phải chờ Server chấp nhận kết nối (nếu ở giao thức TCP) bằng hàm accept trên.  
  
Hàm ACCEPT gồm có 2 thông số:   
- SOCKET s: Socket lắng nghe của SERVER.  
- sockaddr addr: Là cấu trúc sockaddr lưu địa chỉ IP và PORT của CLIENT kết nối tới SERVER.  
- int addrlen: Kích thước cấu trúc địa chỉ IP này.  
  
Hàm ACCPET trả về 1 SOCKET mới  
Socket mới được tạo này đại diện cho 1 Connection (kết nối) mới giữa Server và Client. Sau khi đã truyền dữ liệu thì ta phải đóng SOCKET này lại bằng hàm close như closesocket(connect);  
  
e. **REVC/SEND**   
  
\* Nhận dữ liệu trên giao thức TCP  
  
int recv(  
SOCKET s,   
char FAR\* buf,   
int len,   
int flags  
);  
  
  
\* Gửi dữ liệu giao thức TCP  
  
  
int send(  
SOCKET s,   
const char FAR \* buf,   
int len,   
int flags  
);  
  
  
- SOCKET s: Là SOCKET được tạo ra khi Server chấp nhận kết nối từ CLIENT  
- char FAR\* buf: Là dữ liệu (dạng BYTE – char) nhận hay gửi.  
- int len: Kích thước của dữ liệu.  
- int flags: Một số cờ hiệu đi kèm (thông thường là 0)

 CLOSE/SHUTDOWN  
  
  
int shutdown(  
SOCKET s,   
int how  
);  
  
  
  
int closesocket (SOCKET s);  
  
  
Hủy SOCKET sau một kết nối hoặc kết thúc chương trình.  
Tham số how của shutdown:  
- SD\_RECEIVE: Đóng SOCKET, không cho phép NHẬN nhưng cho phép GỬI.  
- SD\_SEND: Đóng SOCKET, không cho phép GỬI nhưng cho phép NHẬN.  
- SD\_BOTH: Không cho GỬI và NHẬN (giống gọi hàm closesocket).