Lập trình WinSock

- **Chuẩn bị môi trường**
  - Thư viện trực tuyến MSDN
  - Thêm tiêu đề WINSOCK2.H vào đầu mỗi tệp mã nguồn.
  - Thêm thư viện WS32 vào mỗi project bằng cách
    - Dev C++ :
      - Project -> Project Options -> Parameters -> Linker: Add Library or Object (libws2_32.a)
    - Visual studio C++
      - Project -> Property -> Configuration Properties -> Linker -> Input -> Additional Dependencies (WS2_32.LIB)

Lập trình Socket với WinSock

- WinSock
  - Khởi tạo WinSock
  - Giải phóng WinSock
  - Xác định lỗi
- Tạo Socket
- Xác định địa chỉ
- Phân giải tên miền
- Truyền dữ liệu sử dụng giao thức TCP
- Truyền dữ liệu sử dụng giao thức UDP
- Các phương pháp vào ra

1. Khởi tạo WinSock

- WinSock cần được khởi tạo ở đầu mỗi ứng dụng trước khi có thể sử dụng
- Hàm WSAStartup sẽ làm nhiệm vụ khởi tạo

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

Trong đó:

- wVersionRequested: [IN] phiên bản WinSock cần dùng.
- lpWSAData: [OUT] con trỏ chứa thông tin về WinSock cài đặt trong hệ thống.
- Giá trị trả về:
  - Thành công: 0
  - Thất bại: SOCKET_ERROR

Ví dụ khởi tạo WinSock

– Ví dụ

```
WSADATA  wsaData;
WORD wVersion = MAKEWORD(2,2); // Khởi tạo phiên bản 2.2
if (WSAStartup(wVersion,&wsaData))
{
        printf("Version not supported");
}
```

Giải phóng WinSock

– Ứng dụng khi kết thúc sử dụng WinSock có thể gọi hàm sau để giải phóng tài nguyên về cho hệ thống

int WSACleanup(void);

– Giá trị trả về:

- Thành công: 0
- Thất bại: SOCKET_ERROR

Xác định lỗi

- Phần lớn các hàm của WinSock nếu thành công đều trả về 0.
- Nếu thất bại, giá trị trả về của hàm là SOCKET_ERROR.
- Ứng dụng có thể lấy mã lỗi gần nhất bằng hàm

  int WSAGetLastError(void);
- Tra cứu lỗi với công cụ Error Lookup trong Visual Studio -> Tool -> Error Lookup

2. Tạo SOCKET

- Ứng dụng phải tạo SOCKET trước khi có thể gửi nhận dữ liệu
- Hàm **socket** được sử dụng để tạo SOCKET

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

Trong đó:

- **af:** [IN] Address Family, họ giao thức sẽ sử dụng, thường là AF_INET (IPV4)
- **type:** [IN] Kiểu socket, SOCK_STREAM cho TCP/IP và SOCK_DGRAM cho UDP/IP.
- **protocol:** [IN] Giao thức tầng giao vận, IPPROTO_TCP hoặc IPPROTO_UDP

Ví dụ tạo SOCKET

```
SOCKET s1,s2; // Khai báo socket s1,s2

// Tạo socket TCP
s1 = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

// Tạo socket UDP
s2 = socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)
```

Xác định địa chỉ

– WinSock sử dụng **sockaddr_in** để lưu địa chỉ của ứng dụng đích cần kết nối đến

– Ứng dụng cần khởi tạo thông tin trong cấu trúc này

```
struct  sockaddr_in{
    short  sin_family; // Họ giao thức, thường là AF_INET
    u_short  sin_port;  // Cổng, dạng big-endian
    struct  in_addr  sin_addr; // Địa chỉ IP
    char  sin_zero[8];  // Không sử dụng với IPv4
};
```

Xác định địa chỉ

- Sử dụng các hàm hỗ trợ sau:

  - Chuyển đổi địa chỉ IP dạng xâu sang số nguyên 32 bit
    unsigned long inet_addr(const char FAR *cp);
  - Chuyển đổi địa chỉ từ dạng **in_addr** sang dạng xâu
    char FAR *inet_ntoa(struct in_addr in);
  - Chuyển đổi little-endian => big-endian (network order)

    ```
    // Chuyển đổi 4 byte từ little-endian=>big-endian
    u_long htonl(u_long hostlong)
    // Chuyển đổi 2 byte từ little-endian=>big-endian
    u_short htons(u_short hostshort)
    ```
  - Chuyển đổi big-endian => little-endian (host order)

    ```
    // Chuyển 4 byte từ big-endian=>little-endian
    u_long ntohl(u_long netlong)
    // Chuyển 2 byte từ big-endian=>little-endian
    u_short ntohs(u_short netshort)
    ```

Xác định địa chỉ

– Ví dụ: Điền địa chỉ 192.168.0.1:80 vào cấu trúc sockaddr_in

```
SOCKADDR_IN InternetAddr; // Khai báo biến lưu địa chỉ
u_short nPortId = 80; // Khai báo cổng

//Chuyển đổi cổng sang dạng network-byte order và gán cho
//trường sin_port
InternetAddr.sin_port = htons(nPortId);
// Chuyển xâu địa chỉ 192.168.0.1 sang số 4 byte dạng
network-//byte order và gán cho trường sin_addr
InternetAddr.sin_addr.s_addr = inet_addr("192.168.0.1");
```

Phân giải tên miền

- Đôi khi địa chỉ của máy đích được cho dưới dạng tên miền, ứng dụng cần thực hiện phân giải tên miền để có địa chỉ thích hợp.
- Hàm **getnameinfo** và **getaddrinfo** sử dụng để phân giải tên miền
- Cần thêm tệp tiêu đề WS2TCPIP.H vào đầu tệp mã nguồn

```
int getaddrinfo(
    const char FAR *nodename, // Tên miền hoặc địa chỉ cần phân giải
    const char FAR *servname,  // Chuỗi mô tả dịch vụ hoặc cổng
    const struct addrinfo FAR *hints, // Cấu trúc gợi ý
    struct addrinfo FAR *FAR *res  // Kết quả sau khi phân giải
);
```

Giá trị trả về:

- Thành công : 0
- Thất bại: mã lỗi

Phân giải tên miền

– Cấu trúc addrinfo:

```
struct  addrinfo {
        int ai_flags; //AI_PASSIVE / AI_CANONNAME  / AI_NUMERICHOST
        int ai_family; //AF_INET / AF_INET6 / AFUNSPEC
        int ai_socktype; //Loại Socket
        int ai_protocol; //Giai thức giao vận
        size_t ai_addrlen; //Chiều dài của ai_addr
        char *ai_canonname; //Tên miền
        struct sockaddr *ai_addr; //Địa chỉ socket đã phân giải
        struct addrinfo *ai_next; //Con trỏ tới cấu trúc tiếp theo
};
```

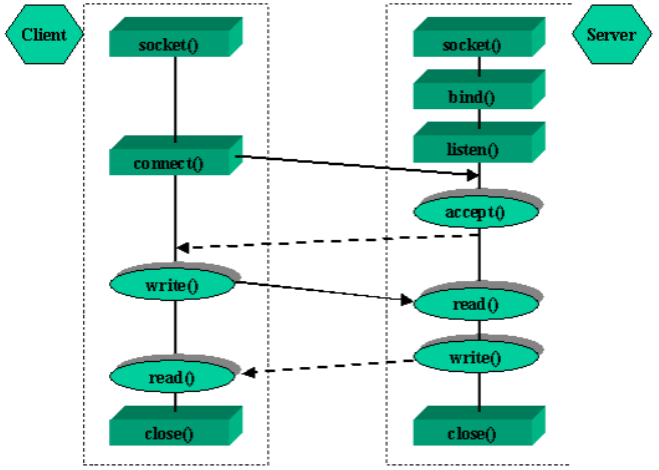
Phân giải tên miền

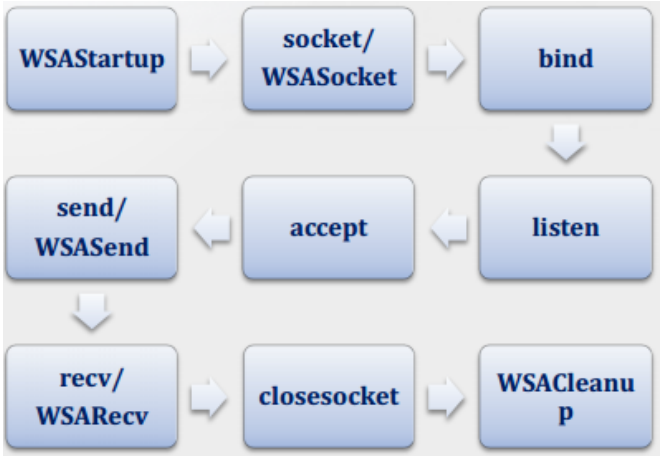
– Đoạn chương trình sau sẽ thực hiện phân giải địa chỉ cho tên miền www.google.com

```
SOCKET s;
struct addrinfo hints, *result;
sockaddr_in      address; // Lưu địa chỉ phân giải được
int rc;
memset(&hints, 0, sizeof(hints));
hints.ai_flags = AI_CANONNAME;
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
rc = getaddrinfo("www.google.com", "http", &hints, &result);
if (rc != 0) {
                // unable to resolve the name
} else {
                memcpy(&address,result->ai_addr,result->ai_addrlen);
}
freeaddrinfo(result);
```

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía Server**
  - Khởi tạo WinSock qua hàm **WSAStartup**
  - Tạo SOCKET qua hàm **socket** hoặc **WSASocket**
  - Gắn SOCKET vào một giao diện mạng thông qua hàm **bind**
  - Chuyển SOCKET sang trạng thái đợi kết nối qua hàm **listen**
  - Chấp nhận kết nối từ client thông qua hàm **accept**
  - Gửi dữ liệu tới client thông qua hàm **send** hoặc **WSASend**
  - Nhận dữ liệu từ client thông qua hàm **recv** hoặc **WSARecv**
  - Đóng SOCKET khi việc truyền nhận kết thúc bằng hàm **closesocket**
  - Giải phóng WinSock bằng hàm **WSACleanup**





Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **bind**: gắn SOCKET vào 1 giao diện mạng của máy

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

Trong đó:

- s: [IN] SOCKET vừa được tạo bằng hàm socket
- name: [IN] địa chỉ của giao diện mạng cục bộ
- namelen: [IN] chiều dài của cấu trúc name

– Thí dụ:

```
SOCKADDR_IN  tcpaddr;
short  port = 8888;
tcpaddr.sin_family = AF_INET; // Socket IPv4
tcpaddr.sin_port = htons(port); // host order => net order
tcpaddr.sin_addr.s_addr = htonl(INADDR_ANY); //Giao diện bất kỳ
bind(s, (SOCKADDR *)&tcpaddr, sizeof(tcpaddr)); // Bind socket
```

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **listen**: chuyển SOCKET sang trạng thái đợi kết nối

  int listen(SOCKET s, int backlog);

  Trong đó:
    - **s: [IN]** SOCKET đã được tạo trước đó bằng hàm socket/WSASocket
    - **backlog: [IN]** chiều dài hàng đợi chấp nhận kết nối

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **accept**: chấp nhận kết nối

SOCKET accept(SOCKET s, struct sockaddr FAR* addr, int FAR* addrlen);

Trong đó:

- **s: [IN]** SOCKET hợp lệ, đã được bind và listen trước đó
- **addr: [OUT]** địa chỉ của client kết nối đến
- **addrlen: [IN/OUT]** con trỏ tới chiều dài của cấu trúc addr. Ứng dụng cần khởi tạo addrlen trỏ tới 1 số nguyên chứa chiều dài của addr

Giá trị trả về là 1 SOCKET mới, sẵn sàng cho việc gửi nhận dữ liệu trên đó. Ứng với mỗi kết nối của client sẽ có một SOCKET riêng.

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **send**: gửi dữ liệu trên SOCKET

int send(SOCKET s, const char FAR* buf, int len, int flags);

Trong đó:

- **s: [IN]** SOCKET hợp lệ, đã được accept trước đó.
- **buf: [IN]** địa chỉ của bộ đệm chứa dữ liệu cần gửi.
- **len: [IN]** số byte cần gửi.
- **flags: [IN]** cờ quy định cách thức gửi, có thể là 0, MSG_OOB, MSG_DONTROUTE

Giá trị trả về:

- Thành công: số byte gửi được, có thể nhỏ hơn **len**
- Thất bại: SOCKET_ERROR

Ví dụ:

```
char szHello[]=”Hello Network Programming”;
send(s,szHello,strlen(szHello),0);
```

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **recv**: nhận dữ liệu trên SOCKET

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

Trong đó:

- **s: [IN]** SOCKET hợp lệ, đã được accept trước đó.
- **buf: [OUT]** địa chỉ của bộ đệm nhận dữ liệu.
- **len: [IN]** kích thước bộ đệm.
- **flags: [IN]** cờ quy định cách thức nhận, có thể là 0, MSG_PEEK, MSG_OOB, MSG_WAITALL

Giá trị trả về:

- Thành công: số byte nhận được, có thể nhỏ hơn **len**
- Thất bại: SOCKET_ERROR

Ví dụ:

```
char buff[100];
int len = 0;
len = recv(s,buff,100,0);
```

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Hàm **closesocket**: đóng kết nối trên một socket

  int closesocket(SOCKET s);

  Trong đó:
    - **s: [IN]** SOCKET hợp lệ, đã kết nối.

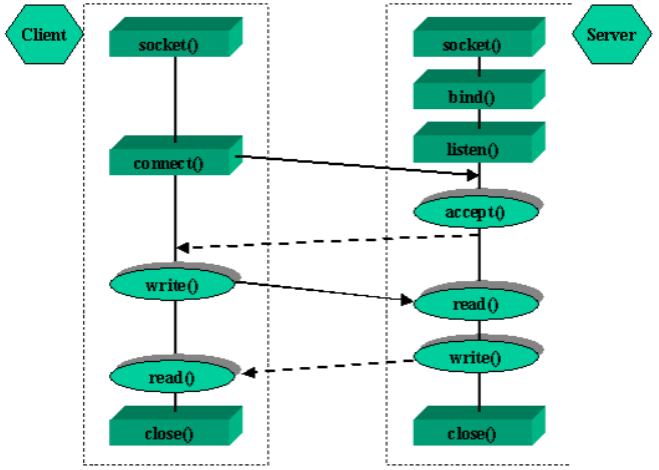
  Giá trị trả về:
    - Thành công: 0
    - Thất bại: SOCKET_ERROR

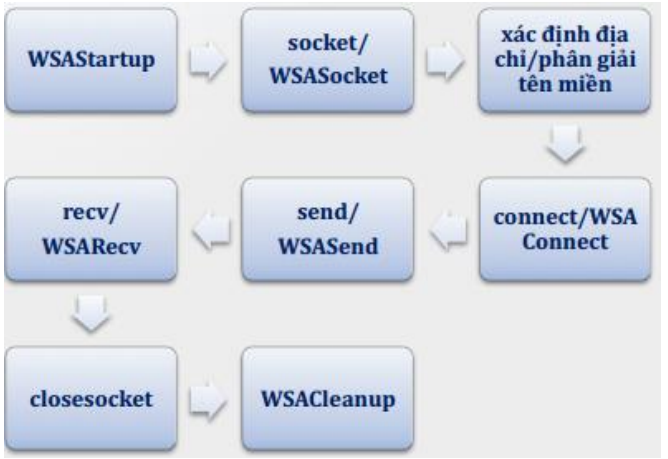
Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía server (tiếp)**
  - Demo : TCPServer

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía Client**
  - Khởi tạo WinSock qua hàm **WSAStartup**
  - Tạo SOCKET qua hàm **socket** hoặc **WSASocket**
  - Điền thông tin về server vào cấu trúc **sockaddr_in**
  - Kết nối tới server qua hàm **connect** hoặc **WSAConnect**
  - Gửi dữ liệu tới server thông qua hàm **send** hoặc **WSASend**
  - Nhận dữ liệu từ server thông qua hàm **recv** hoặc **WSARecv**
  - Đóng SOCKET khi việc truyền nhận kết thúc bằng hàm **closesocket**
  - Giải phóng WinSock bằng hàm **WSACleanup**





Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía Client(tiếp)**
  - Địa chỉ của server xác định trong cấu truc **sockaddr_in** nhờ hàm **inet_addr** hoặc theo **getaddrinfo**
  - Hàm **connect**: kết nối đến server

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

  Trong đó:
    - **s: [IN]** SOCKET đã được tạo bằng **socket** hoặc **WSASocket** trước đó.
    - **name: [IN]** địa chỉ của server.
    - **namelen: [IN]** chiều dài cấu trúc name.

  Giá trị trả về:
    - Thành công: 0
    - Thất bại: SOCKET_ERROR

Truyền dữ liệu sử dụng giao thức TCP

- **Ứng dụng phía Client(tiếp)**
  - Demo: TCPClient

Case study 1

Xây dựng ứng dụng client/server sử dụng **TCP socket** như sau:

- Xây dựng ứng dụng sử dụng phía server tính tổng, hiệu, tích, thương của 2 số nhận được từ client và trả về kết quả cho client.
- Xây dựng ứng dụng sử dụng phía client để gửi giá trị 2 số nguyên đến server, sau đó nhận kết quả và hiển thị lên màn hình.

Truyền dữ liệu sử dụng giao thức UDP

- Ứng dụng không cần phải thiết lập kết nối trước khi gửi tin.
- Ứng dụng có thể nhận được tin từ bất kỳ máy tính nào trong mạng.

Trình tự gửi thông tin bên gửi

Trình tự nhận thông tin bên nhận

Truyền dữ liệu sử dụng giao thức UDP

- **Ứng dụng bên gửi**
  - Hàm **sendto**: gửi dữ liệu đến một máy tính bất kỳ

```
int sendto(
    SOCKET s, // [IN] socket đã tạo bằng hàm socket/WSASocket
    const char FAR * buf,  // [IN] bộ đệm chứa dữ liệu cần gửi
    int len, // [IN] số byte cần gửi
    int flags,  // [IN] cờ, tương tự như hàm send
    const struct sockaddr FAR * to,  // [IN] địa chỉ đích
    int tolen // [IN] chiều dài địa chỉ đích
);
```

Giá trị trả về:

- Thành công: số byte gửi được, có thể nhỏ hơn **len**
- Thất bại: SOCKET_ERROR

Truyền dữ liệu sử dụng giao thức UDP

Đoạn chương trình sau sẽ gửi 1 xâu tới địa chỉ: 202.191.56.69:8888


```
char  buf[]=”Hello Network Programming”; // Xâu cần gửi
SOCKET  sender; // SOCKET để gửi
SOCKADDR_IN receiverAddr; // Địa chỉ nhận
// Tạo socket để gửi tin
sender = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
// Điền địa chỉ đích
receiverAddr.sin_family = AF_INET;
receiverAddr.sin_port = htons(8888);
receiverAddr.sin_addr.s_addr = inet_addr("202.191.56.69");
// Thực hiện gửi tin
sendto(sender, buf, strlen(buf), 0,
(SOCKADDR *)&receiverAddr, sizeof(receiverAddr));
```

Truyền dữ liệu sử dụng giao thức UDP

- **Ứng dụng bên nhận**
  - Hàm **recvfrom**: nhận dữ liệu từ một socket

```
int recvfrom(
    SOCKET s, // [IN] SOCKET sẽ nhận dữ liệu
    char FAR* buf,  // [IN] địa chỉ bộ đệm chứa dữ liệu sẽ nhận được
    int len, // [IN] kích thước bộ đệm
    int flags, // [IN] cờ, tương tự như hàm recv
    struct sockaddr FAR* from,// [OUT] địa chỉ của bên gửi
    int FAR* fromlen // [IN/OUT] chiều dài cấu trúc địa chỉ của bên
    // gửi, khởi tạo là chiều dài của from
);
```

Giá trị trả về:

- Thành công: số byte nhận được
- Thất bại: SOCKET_ERROR

Truyền dữ liệu sử dụng giao thức UDP

– Đoạn chương trình sau sẽ nhận dữ liệu datagram từ cổng 8888 và hiển thị ra màn hình


```
SOCKET  receiver;
SOCKADDR_IN  addr , source;
int len = sizeof(source);
// Tạo socket UDP
receiver = socket(AF_INET, SOCK_DG
// Khởi tạo địa chỉ và cổng 8888
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(INADDR_ANY);
addr.sin_port = htons(8888); // Đợi UDP datagram ở cổng 8888
// Bind socket vào tất cả các giao diện và cổng 8888
bind(receiver,(sockaddr*)&addr,sizeof(SOCKADDR_IN));
```

Truyền dữ liệu sử dụng giao thức UDP

Đoạn chương trình sau sẽ nhận dữ liệu datagram từ cổng 8888 và hiển thị ra màn hình (tiếp):


```
// Lặp đợi gói tin
while (1)
{
    // Nhận dữ liệu từ mạng
    datalen = recvfrom(ListeningSocket,buf,100,0,(sockaddr*)&source,
    &len);
    // Kiểm tra chiều dài
    if (datalen>0)
    {
        buf[datalen]=0;
        printf("Data:%s",buf); // Hiển thị ra màn hình
    }
}
```

Case study 1

Xây dựng ứng dụng client/server sử dụng **TCP socket** như sau:

- Xây dựng ứng dụng sử dụng phía server tính tổng, hiệu, tích, thương của 2 số nhận được từ client và trả về kết quả cho client.
- Xây dựng ứng dụng sử dụng phía client để gửi giá trị 2 số nguyên đến server, sau đó nhận kết quả và hiển thị lên màn hình.