Chương 4. MFC Socket

Trương Đình Huy

Chương 4. MFC Soket

- 4.1. Giới thiệu
- 4.2. CSocket
- 4.3. CAsyncSocket

Chương 4.1 Giới thiệu

- MFC: Microsoft Foundation Classes
- Bộ thư viện hướng đối tượng C++ lập trình ứng dụng trên Window.
- Cung cấp hai lớp hỗ trợ lập trình mạng
  - CAsyncSocket: Đóng gói lại thư viện WinSock dưới dạng hướng đối tượng. Hoạt động ở chế độ bất đồng bộ.
  - CSocket: Kế thừa từ CAsyncSocket và cung cấp giao diện ở mức cao hơn nữa. Hoạt động ở chế độ đồng bộ.
- Hai lớp này không **thread-safe**: đối tượng tạo ra ở luồng nào thì chỉ có thể được sử dụng ở luồng đó.
- Tệp tiêu đề: afxsock.h

Chương 4.2 CSocket

- Khởi tạo thư viện: tự động bởi framework qua hàm AfxSocketInit
- Khởi tạo đối tượng CSocket: Phương thức Create

```
BOOL Create(
  UINT nSocketPort = 0,                  // Cổng, mặc định là 0
  int nSocketType = SOCK_STREAM,         // Kiểu socket
  LPCTSTR lpszSocketAddress = NULL)      // Địa chỉ giao diện mạng, thí dụ
                                         //  “192.168.1.1”
```

Giá trị trả về:

- Khác NULL nếu thành công
- NULL nếu thất bại. Mã lỗi có thể truy nhập qua hàm GetLastError()

Thí dụ:

```
CSocket      Server, Client
Server.Create(8888);
Client.Create();
```

Chương 4.2 CSocket

- Kết nối đến máy khác: Phương thức Connect

```
BOOL Connect(
  LPCTSTR lpszHostAddress,        // Địa chỉ/tên miền máy đích
  UINT nHostPort                  // Cổng
);
BOOL Connect(
  const SOCKADDR* lpSockAddr,     // Địa chỉ máy đích dưới dạng SOCKADDR
  int nSockAddrLen                // Chiều dài cấu trúc địa chỉ
);
```

Giá trị trả về:

- Khác NULL nếu thành công
- NULL nếu thất bại. Mã lỗi có thể truy nhập qua hàm GetLastError()

Thí dụ:

```
CSocket       s;
s.Create();
s.Connect("www.google.com.vn", 80);
```

Chương 4.2 CSocket

- Đợi kết nối từ máy khác: Phương thức Listen

```
BOOL    Listen(
        int nConnectionBacklog = 5 )
```

Giá trị trả về:

- Khác NULL nếu thành công
- NULL nếu thất bại. Mã lỗi có thể truy nhập qua hàm GetLastError()

- Đóng kết nối: Phương thức Close

```
virtual    void    Close( )
```

Chương 4.2 CSocket

- Chấp nhận kết nối từ máy khác: Phương thức Accept

```
virtual BOOL Accept(
  CSocket& rConnectedSocket,        // Socket tương ứng với kết nối mới
  SOCKADDR* lpSockAddr = NULL,// Địa chỉ socket mới dưới dạng SOCKADDR
  int* lpSockAddrLen = NULL         // Chiều dài địa chỉ
);
```

Giá trị trả về:

- Khác NULL nếu thành công
- NULL nếu thất bại. Mã lỗi có thể truy nhập qua hàm GetLastError()

Thí dụ:

```
CSocket       Server, Client;
// Khởi tạo socket Server
…
// Chấp nhận kết nối
Server.Accept(Client);
// Gửi nhận dữ liệu trên Client
…
```

Chương 4.2 CSocket

- Gửi dữ liệu đến máy khác: Phương thức Send

```
virtual int Send(
  const void* lpBuf,      // Bộ đệm chứa dữ liệu cần gửi
  int nBufLen,            // Số byte cần gửi
  int nFlags = 0          // Cờ, chỉ có thể là MSG_OOB nếu có
);
```

Giá trị trả về:

- Số byte gửi được nếu thành công
- SOCKET_ERROR nếu thất bại

Thí dụ:

```
char        buff[]="Hello MFC Socket";
Client.Send(buff,strlen(buff));
```

Chương 4.2 CSocket

- Nhận dữ liệu từ máy khác: Phương thức Receive

```
virtual int Receive(
  void* lpBuf,          // Bộ đệm sẽ nhận dữ liệu
  int nBufLen,          // Kích thước bộ đệm
  int nFlags = 0        // Cờ, có thể là MSG_PEEK hoặc MSG_OOB
);
```

Giá trị trả về:

- Số byte nhận được nếu thành công
- NULL nếu kết nối bị đóng
- SOCKET_ERROR nếu thất bại

Thí dụ:

```
...
char buff[1024];
int buflen = 1024, nBytesReceived;
nBytesReceived = connectedSocket. Receive(buff,1024);
...
```

Chương 4.2 CSocket

- Xây dựng Client bằng CSocket

```
...
CSocket             s;
unsigned char       buff[1024];
char                * request = "GET / HTTP/1.0\r\nHost:www.google.com\r\n\r\n";
int                 len = 0;
s.Create();
s.Connect("www.google.com",80);
s.Send(request,strlen(request));
len = s.Receive(buff,1024);
buff[len] = 0;
printf("%s",buff);
...
```

Chương 4.2 CSocket

- Xây dựng Server bằng CSocket

```
...
CSocket     listen,connect;
Char        * buff = “Hello Network Programming”;
listen.Create(80,SOCK_STREAM,”0.0.0.0”);
listen.Listen();
listen.Accept(connect);
connect.Send(buff,strlen(buff));
connect.Close();
...
```

Chương 4.3 CAsyncSocket

- Đóng gói hoạt động của socket bất đồng bộ
- Nguyên mẫu các hàm vào ra tương tự CSocket nhưng trở về ngay lập tức từ lời gọi.
- Ứng dụng không sử dụng trực tiếp lớp này mà kế thừa và chồng lên các phương thức ảo của lớp để xử lý các sự kiện.
- Các phương thức hay được chồng
  - **OnAccept**: Phương thức này sẽ được gọi mỗi khi có yêu cầu kết nối.
  - **OnClose**: Phương thức này sẽ được gọi mỗi khi socket đầu kia bị đóng.
  - **OnSend**: Phương thức này được gọi khi socket có thể gửi dữ liệu.
  - **OnReceive**: Phương thức này được gọi khi socket nhận được dữ liệu và chờ ứng dụng xử lý
  - **OnConnect**: Phương thức này được gọi khi yêu cầu kết nối được chấp nhận và socket đã sẵn sàng để gửi nhận dữ liệu.

Chương 4.3 CAsyncSocket

- Khởi tạo đối tượng: Phương thức Create

```
BOOL Create(
  UINT nSocketPort = 0,                    // Cổng
  int nSocketType = SOCK_STREAM,           // Kiểu socket
  long lEvent = FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT | FD_CONNECT | FD_CLOSE,
                                           // Mặt nạ sự kiện
  LPCTSTR lpszSocketAddress = NULL         // Địa chỉ socket
);
```

Giá trị trả về :

- Khác NULL nếu thành công
- NULL nếu thất bại

Sự khác biệt duy nhất với CSocket ở phương thức này là tham số lEvent chứa mặt nạ các sự kiện ứng dụng mong muốn nhận được

Chương 4.3 CAsyncSocket

- Xử lý các sự kiện: chồng lên phương thức tương ứng với sự kiện mong muốn

```
void CMyAsyncSocket::OnReceive(int nErrorCode)   // CMyAsyncSocket kế thừa từ
                                                 // AsyncSocket
{
  static int i = 0;
  i++;
  TCHAR buff[4096];
  int nRead;
  nRead = Receive(buff, 4096);
  switch (nRead)
  {
    case 0:
     Close();
     break;
    case SOCKET_ERROR:
     if (GetLastError() != WSAEWOULDBLOCK)
     {
      AfxMessageBox (_T("Error occurred"));
      Close();
     }
     break;
```

Chương 4.3 CAsyncSocket

- Xử lý các sự kiện (tiếp)

```
default:
    buff[nRead] = _T('\0');  // Kết thúc xâu
    CString szTemp(buff);
    m_strRecv += szTemp;   // Chèn xâu nhận được vào cuối m_strRecv
    if (szTemp.CompareNoCase(_T("bye")) == 0)
    {
      ShutDown();
      s_eventDone.SetEvent();
    }
  }
  CAsyncSocket::OnReceive(nErrorCode);
}
```

Chương 4.3 CAsyncSocket

- 3.Viết chương trình HTTP Streaming server thực hiện thao tác sau:
  - Đợi kết nối ở cổng 80
  - Cho phép giới hạn tốc độ upload
  - Xử lý các request từ client gửi đến có dạng

    GET /<TenFile> HTTP/1.1
    Host:
    ....
    \n\n
  - Phản hồi các request như sau:
    - Nếu tên <TenFile> tồn tại trong thư mục hiện tại thì gửi trả phản hồi có dạng

      Status:OK-200\n
      Content-Length:<KichThuocFile>\n
      Content-Type:video/mp4\n
      \n
      \n
      <NoiDungFile>

Chương 4.3 CAsyncSocket

- 3.Viết chương trình HTTP Streaming server thực hiện thao tác sau:
  - Phản hồi các request như sau:
    - Nếu file không tồn tại thì phản hồi lại như sau

      Status: Not Found – 404\n
      Content-Length:<Chieu dai xau phan hoi>\n
      Content-Type:text/html\n
      \n
      \n
      Không tìm thấy tệp tin

Chương 4.3 CAsyncSocket

- 1.Viết chương trình gửi file bằng CAsyncSocket
- 2. Viết chương trình gửi tin nhắn mã hóa qua mạng bằng blocking. Cách thức mã hóa như sau:
  - Server chọn một số nguyên x (0-255) làm mật khẩu. và gửi cho mỗi client khi kết nối đến.
  - Mã ASCII của ký tự được gửi sẽ được cộng thêm x trước khi truyền, bên nhận trừ đi x để hiển thị. Nếu giá trị công thêm >255 thì truyền đi phần dư của giá trị đó khi chia cho 256.