Cơ chế lập trình non-blocking sử dụng WSAAsyncSelect

Như đã đề cập ở các bài trước, nhiều hàm trong lập trình socket hoạt động theo cơ chế blocking ví dụ như hàm accept, hàm recv. Khi server gọi hàm accept(), chương trình server sẽ treo cho đến khi có một client gọi hàm connect() để kết nối đến.

Ở thực tế, chúng ta ít khi nào chấp nhận để chương trình bị treo do đó phải sử dụng một số cơ chế lập trình non-blocking. Trong nội dung bài này, chúng ta sẽ nghiên cứu cơ chế xử lý non-blocking sử dụng WSAAsyncSelect

WSAAsyncSelect sử dụng cơ chế xử lý thông điệp của Windows. Khi windows kiểm tra và nhận thấy có một kết nối đến server sẽ gửi một thông điệp để báo cho server biết, lúc này server mới gọi hàm accept và không phải chờ đợi dẫn đến bị treo chương trình. Tương tự, khi có dữ liệu gửi đến, windows sẽ báo cho ứng dụng. Lúc này ứng dụng mới gọi hàm recv và nhận ngay dữ liệu.

Cú pháp hàm WSAAsyncSelect:

int WSAAsy in in in in	yncSelect(SOCKET , HWND , unsigned int wMsg , long IEvent
); Giải thích ý	nghĩa:
Clai tilicii y	ngina.
S	
socket ca	ần Windows theo dõi giúp
hWnd	
handle to	ới một cửa sổ, windows sẽ gửi các thông điệp báo hiệu đến cửa sổ này.
wMsg	
Loại thô	ng điệp mà Windows sẽ gửi khi có sự kiện xảy ra tại socket. Thông điệp này

lEvent

Cho biết những sư kiên mà windows sẽ giám sát.

Giá trị IEvent thường là kết hợp của các giá trị FD_READ, FD_WRITE, FD_CLOSE, FD_ACCEPT.

thường sẽ là thông điệp tư đinh nghĩa, có giá tri lớn hơn WM USER

Ví dụ lEvent có thể là FD_READ | FD_CLOSE | FD_ACCEPT Các sư kiên chính:

Sự kiện	Ýng	hĩa						
FD_READ	Báo	hiệu	có	dữ	liệu	gửi	đến	cho

Sự kiện	Ý nghĩa							
	socket, sẵn sàng để gọi hàm recv							
FD_ACCEPT	Báo hiệu cho socket server (đã gọi hàm listen) có một kết nối từ phía client							
FD_CLOSE	Báo hiệu đầu còn lại của kết nối đã đóng socket							

Môt số đặc điểm cần lưu ý:

- Windows ngay lập tức đặt socket s vào trạng thái non-blocking, bất chấp giá tri của lEvent
- Các socket cha nếu đã được gọi hàm WSAAsyncSelect trước đó, socket con sinh ra bởi hàm accept sẽ có cùng đặc điểm với socket cha.
- Lời gọi hàm WSAAsyncSelect sau sẽ ghi đè lên lời gọi hàm trước, không có ý nghĩa kết hợp

Trong bất kỳ thông điệp (message) nào của Windows đều có 2 thông số quan trọng là wParam và IParam. Với sự kiện sinh ra bởi WSAAsyncSelect thì: wParam: là socket có sư kiên phát sinh

IParam: hai byte thấp (low word) cho biết sự kiện phát sinh (FD_READ, FD_ACCEPT...), hai byte cao (high word) cho biết lỗi phát sinh nếu có. Để thao tác dễ dang trên IParam, windows đã đinh nghĩa sẵn 2 macro:

```
#define WSAGETSELECTERROR(lParam)
                                              HIWORD(lParam)
#define WSAGETSELECTEVENT(lParam)
                                              LOWORD (1Param)
Nguyên tắc khai báo và xử lý sư kiên
Giả sử ta đặt tên sự kiên cần bắt là WM SOCKET, đầu tiên ta sẽ định nghĩa
#define WM_SOCKET WM_USER+1
gọi hàm WSAAsyncSelect:
WSAAsyncSelect(socket,m hWnd,WM SOCKET,FD_ACCEPT|FD_READ|
FD CLOSE);
Có 2 cách để "bắt" và xử lý sự kiện này.
Cách 1:
Trước hết xây dựng hàm để xử lý sự kiên:
void CChatDlg::SockMsg(WPARAM wParam, LPARAM IParam)
     if (WSAGETSELECTERROR(IParam))
     {
            // Có lỗi xảy ra, xuất th lỗi và ngắt socket
           closesocket(wParam);
     switch(WSAGETSELECTEVENT(IParam))
           case FD ACCEPT:
                 //Có 1 kết nối đến, gọi hàm accept để tiếp nhân
```

sockClient = accept(wParam,NULL,NULL);

```
break;
              case FD READ:
                      //Có dữ liệu gửi đến, gọi hàm recv
              case FD CLOSE:
                      //Đầu bên kia đã đóng socket
                      closesocket(wParam);
                      break;
       }
Vấn đề còn lại, phải báo hiệu cho windows biết khi có sự kiện WM SOCKET sẽ
gọi giúp hàm SockMsq.
Đế làm việc này, ta có thể kế thừa hàm WindowProc sử dụng Class Wizard:
MFC ClassWizard
                                                                                ? | X |
                 Member Variables
   Message Maps
                                 Automation | ActiveX Events |
  Project:
                                    Class name:
                                                                      Add Class.
  SimpleChatServer
                                    CSimpleChatServerDlg
                                                                 •
                                                                       Add Function
  C:\...\SimpleChatServerDlg.h, C:\...\SimpleChatServerDlg.cpp
  Object IDs:
                                    Messages:
                                                                      Delete Function
  CSimpleChatServerDlg
                                     PreCreateWindow
                                                                 ٠
                                                                         Edit Code
  IDC_EDITCHAT
                                     PreSubclassWindow
   IDC_EDITMSG
IDC_LISTEN
IDC_SEND
                                     PreTranslateMessage
                                     Serialize
                                     WindowProc
   IDQŪIT
                                     WinHelp
                                    WM CANCELMODE
  Member functions:
   W OnSend
                          ON_IDC_SEND:BN_CLICKED
   W OnSysCommand
                          ON_WM_SYSCOMMAND
   V PreTranslateMessage
      WindowProc
               Dispatches messages through the message map
  Description:
                                                                             Cancel
Hàm WindowProc sẽ như sau:
              CChatDlg::WindowProc(UINT
LRESULT
                                                  message.
                                                                WPARAM
                                                                               wParam.
LPARAM IParam)
{
       // TODO: Add your specialized code here and/or call the base class
       if (message==WM SOCKET){
              SockMsg(wParam,IParam);
```

Như vậy, khi có sự kiện VM_SOCKET phát sinh, hàm SockMsg sẽ được gọi.

return CDialog::WindowProc(message, wParam, IParam);

}

Cách 2:

```
Xây dựng hàm SockMsg như trên, nhưng kiểu trả về của hàm là LRESULT như
LRESULT CChatClientDlg::SockMsg(WPARAM wParam, LPARAM IParam)
       if (WSAGETSELECTERROR(IParam))
       {
               // Display the error and close the socket
              closesocket(wParam);
       switch(WSAGETSELECTEVENT(IParam))
              case FD READ:
                      break;
              case FD CLOSE:
                     break;
              ......
       return 0;
Sau
                                           ON MESSAGE
                                                                nàm
                                                                         giữa
        đó
               khai
                        báo
                                                                                  đoạn
                                macro
BEGIN_MESSAGE_MAP và END_MESSAGE_MAP như sau:
                            DDX_Text(pDX, IDC_SERVADD, m_servAdd);
//}}AFX_DATA_MAP
boutDlg
hatClientApp
hatClientDlg
bals
                         BEGIN_MESSAGE_MAP(CChatClientDlg, CDialog)
                            //{{AFX_MSG_MAP(CChatClientDlg)
                            ON_WM_SYSCOMMAND()
                            ON WM PAINT()
                            ON_WM_QUERYDRAGICON()
ON_BN_CLICKED(IDCONNECT, OnConnect)
                            ON_MESSAGE(WM_SOCKET, SockMsg)
                            ON_BN_CLICKED(IDC_SEND, OnSend)
//}}AFX_MSG_MAP
                         END_MESSAGE_MAP()
```

Những phần trình bày ở trên là những nét chính của lập trình sử dụng WSAAsyncSelect. Đề nghị các bạn nên đọc thêm trong sách Network Programming và MSDN cũng như xem thêm ví dụ.

// CChatClientDlg message handlers
BOOL CChatClientDlg::OnInitDialog()