Winsock API Explained

فهرست مطالب

اخطار

قبل از شروع باید بگم که یادگیری توابع Winsock API چیز آسونی نیست.من شدیدا توصیه میکنم که برای مطالعه این مقاله و بکار بستن موارد ذکر شده حد اقل دو سال سابقه برنامه نویسی با VB رو داشته باشید.همچنین شدیدا توصیه میکنم که با مفاهیم چند ریسمانی (Multi-threading) و SubClassing Messages آشنا باشید و در آخر در فهم و استفاده از توابع کتابخانه ای Win32API مشکلی نداشته باشید.

مقدمه

مطمئن هستم اگر در حال خوندن این مقاله هستید به این معنا است که چند بار با کنترل Winsock در طول زندگی تون سر و کار داشتین.صدها برنامه چت از جمله IRC clients/servers و صدها برنامه FTP clients/servers و صدها برنامه دیگه وجود دارن که از کنترل Winsock استفاده میکنن.مقالات و نمونه کدهای زیادی روی اینترنت در سایت های آموزشی و فروم های مختلف وجود داره که طرز کار با کنترل Winsock رو توضیح دادن.اما چند نفر از این مردم دقیقا میدونن و میفهمن که چه چیزی زیر سطح ظاهری این کنترل رخ میده ؟ خوب، مسلما بعد از انتشار همچین مقاله ای بیشتر از قبل خواهند فهمید.

راه اندازی و پاکسازی

با توابع Winsock API شما می بایست یک تابع رو صدا بزنید تا توابع API برای بار اول بارگذاری و اجرا بشن و از تابع دیگری هم باید برای خاتمه دادن به این اجرا استفاده کنید.در کل هر اجرا (initialization) می بایست یک تابع زوج برای خاتمه دادن (Terminate) به خودش داشته باشه.کدهای زیر رو به پروژه اضافه کنید که برای اجرای اولیه و خاتمه دادن به پروسه توابع استفاده میشن.

```
Private Declare Function WSAStartup Lib "ws2_32.dll" (ByVal wVR As Long, lpWSAD
As WSAData) As Long
Private Declare Function WSACleanup Lib "ws2_32.dll" () As Long

Private Const WSADESCRIPTION_LEN = 257
Private Const WSASYS_STATUS_LEN = 129

Private Const SCK_VERSION1 = &H101                ‘Windows sockets version 1.1
Private Const SCK_VERSION2 = &H202                ‘Windows sockets version 2.2

Private Type WSAData
    wVersion      As Integer                              'Version
    WHighVersion  As Integer                              'High Version
    szDescription       As String * WSADESCRIPTION_LEN    'Description
    szSystemStatus      As String * WSASYS_STATUS_LEN     'Status of system
    iMaxSockets         As Integer                        'Maximum number of sockets allowed
    iMaxUdpDg           As Integer                        'Maximum UDP datagrams
    lpVendorInfo        As Long                           'Vendor Info
End Type
```

این تعریف برای آماده سازی و خاتمه دادن به توبعه Winsock API لازم هستند.برای آماده سازی و اجرای توابع Winsock API شما از تابع WSAStartup استفاده میکنید.پارامتر اول یکی از دو مقدار SCK_VERSION1 و یا SCK_VERSION2 هست و پارامتر دوم متغیری از نوع داده WSAData می باشد.یکبار که این تابع رو اجرا کردید متغیر نوع داده WSAData که به تابع پاس داده شده بود، با اطلاعات مفیدی درباره توابع فعلی در حال اجرا پر میشه.برای خاتمه دادن به اجرای این توابع خیلی ساده از تابع WSACleanup استفاده کنید.نکته قابل توجه و بسیار مهم درباره این دو تابع این هست که هر تابع فقط یکبار باید اجرا بشن.

تغییر طبقه بندی بایت ها

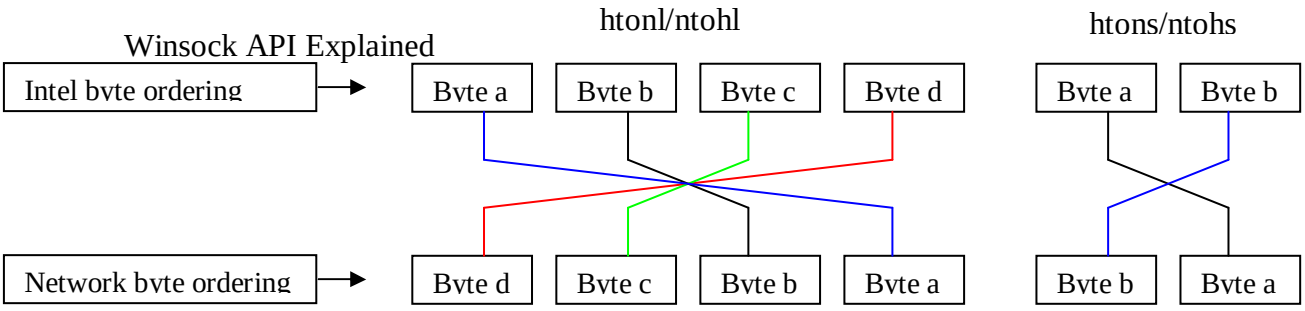
کامپیوترهای اینتل و پروتوکل های شبکه از طبقه بندی معکوس بایت ها نسبت به یکدیگر استفاده میکنن.برای روشن تر کردن این مساله، کامپیوتر اینتل بایت اول رو میبینه و به عنوان بایت اول در نظر گرفته میشه در حالی که این بایت اول در پروتوکل های شبکه بایت آخر هست.بنابراین هر بار که نیاز داریم تا مقداری عددی رو بفرستیم مثل شماره پورت، نیاز داریم ترتیب دهی بایت ها رو به روش طبقه بندی پروتوکول های شبکه تبدیل کنیم.چهار تابع API وجود دارن که این کار رو برای ما انجام میدن.

```
‘htons – host to network short
Private Declare Function htons Lib "ws2_32.dll" (ByVal hostshort As Integer) As Integer

‘htonl – host to network long
Private Declare Function htonl Lib "ws2_32.dll" (ByVal hostlong As Long) As Long

‘ntohs – network to host short
Private Declare Function ntohs Lib "ws2_32.dll" (ByVal netshort As Integer) As Integer

‘ntohl – network to host long
Private Declare Function ntohl Lib "ws2_32.dll" (ByVal netlong As Long) As Long
```



پورت ها

پورت ها بسیار ساده هستند.تجسم کنید که هر اتصالی به کامپیوتر شما نخی است از یک سیم تلگراف.و این اتصال خودش نسبت به وب همین حالت رو داره.MSN Messenger از نخ دیگه ای استفاده میکنه،IRC از یه نخ دیگه و الی آخر.در این قالب، پورت محلی شما آخر ریسمانی است که به کامپیوتر شما متصل شده و پورت راه دور میزبان آخر ریسمان کامپیوتری است که شما به آن متصل شده اید.هنگام کار کردن با پورت ها در Winsock باید دقت داشته باشیم که بایت های شماه پورت ها رو به روش طبقه بندی پروتوکول های شبکه تبدیل کنیم.با استفاده از توابعی که در بالا نوشته شدند.هر پورت شماره مخصوص به خودش رو داره.برای مثال پورت ۸۰ برای اتصال به صفحه های وب در نظر گرفته شده، پورت ۲۱ بطور معمول برای پروتوکول FTP در نظر گرفته شده و پورت ۲۵ اکثر اوقات برای پروتوکل SMTP (Simple Mail Transfer Protocol) استفاده میشه.

ایجاد کردن و بستن سوکت ها

اگر با مقایسه قبلی ادامه بدیم، یک سوکت محیطی است که آخر ریسمان ها (threads) میتونن بهش ضمیمه بشن.اولین کاری که باید بکنیم ایجاد یک سوکت هست.در واقع هر سوکت تنها عددی است که به هر اتصال نسبت داده میشه.بیشتر توابع Winsock API به این عدد نیاز دارن تا بدونن کدوم اتصال رو باید دستکاری کنن. میتونید این شماره ها رو چیزی شبیه به هندل پنجره های ویندوز بدونید (اگر زیاد باهاشون سر و کار دارید).شما با استفاده از تابع Socket یک سوکت جدید ایجاد میکنید.همچنین صدا زدن این تابع تمام حالات و وضعیت ها رو باطل میکنه (مثل گوش دادن روی یک پورت) چرا که چطور یک سوکت میتونه وضعیتی داشته باشه وقتی که حتی دیگه وجود نداره ؟ همچنین صدا زدن این تابع باعث میشه تا تمام اتصالاتی که روی اون سوکت بنا شده بودن از بین برن.برای انجام دادن این دو وظیفه ساده شما به تعاریف جدیدی که در زیر اومدن نیاز خواهید داشت.

```
Private Declare Function socket Lib "ws2_32.dll" (ByVal af As Long, ByVal s_type As Long
, ByVal Protocol As Long) As Long
Private Declare Function closesocket Lib "ws2_32.dll" (ByVal s As Long) As Long

'Windows Socket types
Private Const SOCK_STREAM = 1    'Stream socket

'Address family
Private Const AF_INET = 2        'Internetwork: UDP, TCP e.t.c

'Socket Protocol
Private Const IPPROTO_TCP = 6    'TCP
```

پارامتر اول تابع Socket خانواده آدرس ها هست. این به Winsock کمک میکنه تا خانواده آدرس ها رو تفسیر کنه و بفهمه.تا وقتی که کارهای پیشرفته تری نسبت به موارد بحث شده در این مقاله نمی پردازین مقدار AF_INET رو بی این پارامتر پاس بدین.این مقدار به Winsock میفهمونه که این سوکت به خانواده آدرس ها در اینترنت مربوط میشه.

پارامتر دوم نوع سوکت رو تعیین میکنه. در مجموع سه نوع از سوکت ها وجود دارن : Raw, Stream, Datagram. نوع سوکت Stream برای پروتوکل هایی مثل TCP و داده های Stream استفاده میشه.نوع سوکت Datagram در پروتوکل هایی مثل UDP استفاده میشن یعنی جایی که داده های در پاکت های انتشار پیدا میکنن.و در آخر نوع سوکت Raw برای آزمایش و تست کردن پروتوکل های جدید استفاده میشه که به برنامه نویسان اجازه میده تا نوع پاکت های داده خودشون رو طراحی کنن.برای هدف و مقصود این مقاله (که تنها TCP/IP رو آموزش میده) ما به تابع Socket مقدار SOCK_STREAM (Stream Socket) رو پاس میدیم.و پارامتر سوم پروتوکل مورد استفاده هست.ما به سادگی مقدار IPPROTO_TCP (TCP/IP protocol) رو به تابع پاس میدیم.این تابع مقداری از نوع Long رو برمیگردونه.این یک مقدار منحصر به فرد هست که شما با استفاده از اون میتونید تمام اتصال ها به کامپیوترتون رو شناسایی و دستکاری کنید.چیزی که سعی می کردم از سوکت ها توضیح بدم رو الان باید بهتر متوجه شده باشین.

```
Let lngSocket = Socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
Call CloseSocket (lngSocket)
```

اتصال به میزبان راه دور

برای وصل شدن به یک سرور ما ابتدا باید یک ساختار رو از اطلاعات پر کنیم که پورت مورد نظر برای اتصال، آدرس اتصال و خانواده آدرس ها رو تشریح میکنه.بعد از آن ما خیلی ساده تابع Connect رو صدا میزنیم.تعاریف جدید رو در زیر میبینید.

```
Private Declare Function Connect Lib "ws2_32.dll" Alias "connect" (ByVal s As Long,
 ByRef name As SOCKADDR_IN, ByVal namelen As Long) As Long

'Socket Address structure
Private Type SOCKADDR_IN
    sin_family          As Integer        ‘Address family
    sin_port            As Integer        ‘Port
    sin_addr            As Long           'Long address
    sin_zero(1 To 8)    As Byte           ‘Not used by us
End Type
```

پارامتر اول تابع Connection یعنی پارامتر "s" به سوکت ما اشاره داره.این همون مقدار از نوع Long هست که توسط تابع Socket برگشت داده شد. پارامتر دوم ساختار SOCKADDR_IN و پارامتر سوم طول متغیری از نوع SOCKADDR_IN هست.

پر کردن این ساختار با اطلاعات مفید و قابل استفاده به سادگی که فکر میکنید نیست. فیلد Sin_family همان مقداری است که پیشتر در سوکت خود استفاده کردید که در اینجا AF_INET هست. فیلد sin_port همان شماره پورت اتصال مورد نظر هست، اما دقت داشته باشید که باید با استفاده از تابع "htons" طبقه بندی بایت ها رو به روش پروتوکل های اینترنت تبدیل کنید. فیلد sin_addr مقداری از نوع Long هست در نوع طبقه بندی بایت ها به روش شبکه که آدرس میزبان رو نمایش میده. این فیلد رو در ادامه توضیح خواهم داد. پارامتر آخر برای استفاده های داخلی هست که اینجا از بحث خارجه.

خوب، چطور ما آدرس میزبان راه دور از نوع داده Long رو بدست میاریم ؟
خوب، ما میتونیم از توابع زیر استفاده کنیم.

```
Private Declare Function gethostbyname Lib "ws2_32.dll" (ByVal host_name As String)
As Long
Private Declare Function inet_addr Lib "ws2_32.dll" (ByVal cp As String) As Long
```

ما میتونیم به تابع GetHostByName نام یک وب سایت رو پاس بدیم، برای مثال www.microsoft.com و یا نام یک کامپیوتر در شبکه مثل "Station85". هرچند اگر قصد تبدیل یک آدرس IP مثل "127.0.0.1" رو به نام میزبان داشتید میتونید از تابع inet_addr استفاده کنید. هر دوی این توابع مقداری از نوع داده Long رو برمیگردونن. توجه داشته باشید که هر دوی این توابع مقادیر داده Long رو به روش پروتوکول های شبکه طبقه بندی میکنند بنابراین نیازی به تبدیل طبقه بندی بایت ها نیست.

ایفای نقش سرور ها

به تعریفی ساده، یک سرور برای کلاینت هایی که قصد اتصال به کامپیوتر رو دارن گوش میکنه، بعد از اون این درخواست ها رو میپذیره و برای هر کدوم یک اتصال بنا میکنه.

برای گوش کردن برای درخواست های اتصال باید تابع Listen رو صدا بزنیم. این اولین باری است که قصد دارم به توابع مسدود کردن هم اشاره داشته باشم. متد Listen برنامه رو در حالت عادی قفل (freeze) خواهد کرد تا وقتی که درخواست اتصالی شنیده بشه و تابع مقداری رو برگردونه. این اتفاق برای توابع دیگری هم مثل Connect,GetHostByName رخ میده اگرچه این توابع مقدار زمان نامشخصی رو برای این قفل کردن برنامه به اجرا میذارن بنابراین لازم بود تا بهشون اشاره کنم. روشی وجود داره تا بتونیم توابع رو مجبور کنیم تا مقادیر بازگشتی رو بصورت غیر همزمان (asynchronous) برگشت بدن. اما فعلا این از بحث خارجه و در ادامه توضیح خواهم داد. بیشتر سرور ها که از نوع مسدود کردن (blocking mode) استفاده میکنن متد Listen رو در یک حلقه بی نهایت و در یک ریسمان (thread) دیگه اجرا میکنن (خیلی گیج نشین اینجا از ریسمان های سخت افزاری و حافظه حرف نمیزنم، منظور ریسمان ها در سطح نرم افزار هست).

به تعاریف زیر نیاز خواهید داشت :

```
'Server side Winsock API functions
Private Declare Function Bind Lib "ws2_32.dll" Alias "bind" (ByVal s As Long,
ByRef name As SOCKADDR_IN, ByRef namelen As Long) As Long

Private Declare Function Listen Lib "ws2_32.dll" Alias "listen" (ByVal s As Long,
 ByVal backlog As Long) As Long

Private Declare Function Accept Lib "ws2_32.dll" Alias "accept" (ByVal s As Long,
 ByRef addr As SOCKADDR_IN, ByRef addrlen As Long) As Long

Private Const SOMAXCONN = &H7FFFFFFF
```

قدم اول پر کردن ساختار SOCKADDR_IN هست درست مثل موقعی که میخواستیم به یک میزبان راه دور وصل بشیم. در اینجا مقدار فیلد sin_port شماره پورتی هست که میخوایم روی اون به درخواست های اتصال گوش کنیم و مقدار فیلد sin_addr آدرسی هست که میخوایم دربارش آگاه باشیم. اگر میخواهید تمام درخواست های اتصال رو دریافت کنید یه سادگی مقدار این پارامتر رو "0.0.0.0" پاس بدین. تعاریف مربوط به inet_addr در قسمت اتصالات قبلا توضیح داده شد.

قدم بعدی صدا زدن تابع Bind هست. این تابع به منظور مقید کردن یک سوکت به یک آدرس IP استفاده میشه.این تابع دقیقا پارامترهای مشابه تابع Connect رو داره. در آخر تابع Listen رو صدا بزنید. الان دیگه باید بدونید که پارامتر "s" همون سوکت ما هست. برای پارامتر دوم که backlog نام گرفته به سادگی مقدار ثابت SOMAXCONN رو پاس بدین. وقتی که تابع Listen مقداری رو برگشت داد بدین معنا است که در انتظار دریافت یک درخواست اتصال هستیم.

برای پذیرفتن درخواست اتصال تابع Accept رو صدا بزنید. در اینجا هم تمام پارامتر ها مشابه پارامتر های توابع Bind, Connect هستن اما اینبار لازم نیست تا ساختار مورد نظر رو با چیزی پر کنید. تنها یک متغیر از نوع ساختار

ذکر شده بسازید و به همون شکلی که خالی هست به تابع ارجاع بدین. تابع Accept یک مقدار هندل سوکت جدید برمیگردونه که مربوط به کامپیوتری هست که درخواست اتصالش پذیرفته شده. این مقدار رو ذخیره کنید و بعد میتونید از اون در توابعی مثل Send, Recv استفاده کنید که در ادامه توضیح داده شدن.

فرستادن و دریافت داده ها

خوشبختانه این بخش ساده است. دو تابع برای فرستادن و دریافت داده ها وجود دارن (در واقع دو تابع دیگه هم وجود دارن که در پروتوکول هایی مثل UDP استفاده میشن به نام های SendTo, RecvFrom که در این مقاله نمیگنجه).

```
'Data transfer functions
Private Declare Function Recv Lib "ws2_32.dll" Alias "recv" (ByVal s As Long,
ByRef buf As Any, ByVal buflen As Long, ByVal Flags As Long) As Long

Private Declare Function Send Lib "ws2_32.dll" Alias "send" (ByVal s As Long,
ByRef buf As Any, ByVal buflen As Long, ByVal Flags As Long) As Long
```

برای فرستادن داده به سادگی تابع Send رو صدا بزنید. به طور معمول پارامتر s سوکت متصل شده هست، پارامتر buf یک مقدار رشته ای یا آرایه ای از بایت ها هست و پارامتر buflen طول داده هست. اگرقصد فرستادن داده رشته ای رو دارید با تابع Len() میتونید طول رشته رو بگیرید و اگر داده آرایه ای از بایت ها هست میتونید از توابع LenB(), UBOUND استفاده کنید. تابع Send مقداری رو بر میگردونه که نمایانگر تعداد بایت های فرستاده شده هست.

برای دریافت داده ها شما یک آرایه از نوع بایت تعریف میکنه و به پارامتر buf مربوط به Recv ارجاع میدین. پارمتر buflen هم مطعاقبا طول آرایه بایت ها هست. تابع Recv تعداد بایت های دریافت شده رو برمیگردونه و در صورت اینکه هیچ داده ای در بافر Winsock نباشه مقدار 0 رو برمیگردونه.

*سوکت های غیر همزمان (**Asynchronous Sockets**)*

اینجا جایی است که یکمی مباحث سخت میشن و مباحثی مثل Subclassing هم مداخله میکنن. پیشتر ما از سوکت های مسدود کننده (blocking sockets) استفاده کردیم. این بدین معناست که توابع مقداری رو برنمیگردونن تا وقتی که وظیفه ای بطور کامل انجام بشه. برای مثال اگر قرار باشه تا نام یک میزبان راه دور رو شناسایی کنیم، تابع برنامه رو برای چند ثانیه ای قبل از برگشت دادن مقدار، قفل میکنه و این روی کاربران شما اثر خوشایندی نداره که سوکت های روی همان ریسمان (thread) اجرا بشن که برنامه اصلی داره اجرا میشه.

اما چیزی وجود داره به اسم سوکت های غیر همزمان یا Asnych Sockets که میتونیم برای رفع همچین مشکلی ازش استفاده کنیم. هنگام استفاده از این نوع سوکت ها ما فورا از مقادیر بازگشتی توابع آگاه میشیم که این کار با استفاده از سیستم Subclassing Messages انجام میشه که احتمالا (یا حتما!) باید بدونید که این به چه معنا است. من اینجا نمیتونم کامل Subclassing رو توضیح بدم و فرض بر این هست که شما برنامه یا پنجره های مورد نظر خودتون رو به درستی Subclassing میکنید. برای اطلاعات بیشتر و آشنایی با این مبحث میتونید به سایت هایی مثل بابا گوگل و یا PSCode.com مراجعه کنید.

تغییر طریقه سوکت ها به سوکت های غیر همزمان

برای تغییر دادن طریقه بنا شدن یک سوکت به سوکت غیر همزمان از تابع API جدیدی بنام WSAAsyncSelect استفاده میکنیم. در زیر مقداری دیگه از تنظیمات کامل رو میبینید (لیست وحشتناک و عظیمی از این تعاریف و تنظیمات برای استفاده کامل از Winsock API هست که باید اعمال بشن!).

```
Private Declare Function WSAAsyncSelect Lib "ws2_32.dll" (ByVal s As Long,
 ByVal hwnd As Long, ByVal wMsg As Long, ByVal lEvent As Long) As Long

'Winsock messages that will go to the window handler
Private Enum WSAMessage
        FD_WRITE = &H2&                 'Data is ready to be written to the buffer
        FD_READ = &H1&                  'Data is ready to be read from the buffer
        FD_CONNECT = &H10&              'Connection esatblished
        FD_CLOSE = &H20&                'Connection closed
        FD_ACCEPT = &H8&                'Connection request pending
End Enum
```

پارامتر s باز هم بطور معمول همان سوکت ما است. در این جا سوکت هنوز متصل نشده و در هیچ وضعیتی (مثل گوش کردن برای درخواست های اتصال) هم قرار نداره. پارامتر hwnd هندل پنجره ای هست که شما Subclassed کردین.پارامتر wMsg شماره پیغامی هست که میل دارید در صورت وجود Winsock به شما اطلاع بده. این میتونه هر نوع پیغامی باشه. فقط دقت داشته باشید که شماره های استفاده شده شماره هایی نباشن که قبلا توسط Winsows Messages Subclassing استفاده شدن و یا مقدار غلطی رو به شما خواهند داد. پارامتر lEvents رویداد هایی هست که مایل هستید درباره آنها آگاه بشید. میتونید هر رویداد دلخواهی رو استفاده کنید فقط دقت کنید که در ساختار شمارش یعنی WSAMessage Enum واردش کرده باشید.میتونید از چند رویداد به شکل زیر استفاده کنید :

```
lEvents = FD_CONNECT Or FD_CLOSE Or FD_READ
```

حالا دفعه بعدی که تابع Connect رو صدا بزنید مستقیما پیغامی به کنترل کننده پیغام های شما ارسال خواهد شد.

شناسایی میزبان راه دور بصورت غیر همزمان

اگر قصد دارید یک آدرس IP رو شناسایی کنید میتونید از تابع inet_addr استفاده کنید. اگرچه اگر بخواهید نام یک میزبان راه دور رو بصورت www.microsoft.com شناسایی کنید و مقدار نوع داده Long اون رو بگیرید همونطور که پیشتر گفته شد از تابع GetHostByName استفاده میکنید. این یک تابع مسدود کننده هست و از آنجایی که مستقیما با سوکت ها کار نمی کنه، تغییر دادن طریقه کار یک سوکت به غیر همزمان باعث غیر همزمان شدن این تابع نمیشه. در عوض ما نسخه دیگری از این تابع داریم که بصورت غیر همزمان یا Async کار میکنه :

```
Private Declare Function WSAAsyncGetHostByName Lib "ws2_32.dll" (ByVal hwnd As Long,
 ByVal wMsg As Long, ByVal strHostName As String, buf As Any, ByVal buflen As Long)
As Long
```

پارامتر اول هندل پنجره ای است که شما Subclassed کردین و پارامتر دوم پیغام Winsock شما هست. این همان مقداری است که شما پیشتر ایجاد کردین و به تابع WSAAsyncSelect پاس دادین. پارامتر strHostName نام میزبان راه دوری است که قصد شناسایی اش رو دارید. پارامتر buf بطور معمول آرایه ای از بایت هست تا مقدار برگشتی رو ذخیره کنه و پارامتر buflen هم طول این آرایه هست که گفتیم میتونید با توابع Len(), LenB(), UBOUND طول آرایه یا متغیر رشته ای رو بدست بیارین.

گرفتن اطلاعات سوکت ها

این بخش کوچک آخر درباره گرفتن اطلاعات درباره سوکت های محلی و یا راه دور هست. ما میتونیم اطلاعاتی درباره آدس IP، پورت ها و نام سرورهای هم محلی و هم راه دور رو بدست بیاریم. چطور این کارها رو انجام میدیم ؟ با یک سری توابع جدید :

```
Private Declare Function getsockname Lib "ws2_32.dll" (ByVal s As Long, ByRef name As
SOCKADDR_IN, ByRef namelen As Long) As Long

Private Declare Function getpeername Lib "ws2_32.dll" (ByVal s As Long, ByRef name As
SOCKADDR_IN, ByRef namelen As Long) As Long

Private Declare Function gethostbyaddr Lib "wsock32.dll" (haddr As Long,
 ByVal hnlen As Long, ByVal addrtype As Long) As Long

'HostEnt Structure
Private Type HOSTENT
    hName         As Long       'Host Name
    hAliases      As Long       'Alias
    hAddrType     As Integer    'Address Type
    hLength       As Integer    'Length
    hAddrList     As Long       'Address List
End Type
```

برای گرفتن اطلاعات از کامپیوتر محلی تابع GetSockName و برای کامپیوتر های راه دور (Remote) تابع GetPeerName را صدا میزنیم. پارامتر اول سوکت متصل شده هست. پارامتر دوم متغیری از نوع داده SOCKADDR_IN هست که قبلا توضیح داده شد و پارامتر سوم طول متغیر داده SOCKADDR_IN هست که با تابع Len() بدست میاد. حالا تمام اطلاعات در این ساختار ارجاع داده شده وجود داره (توجه داشته باشید که شماره پورت ها رو با تابع ntohs به روش پروتوکول های شبکه تدبیل کنید). شما به سادگی میتونید آدرس طولانی از نوع Long رو به IP تبدیل کنید و همچنین میتونید به نام میزبان تبدیلش کنید با استفاده از تابع GetHostByAddr. پارامتر اول آدرس است، به پارامتر دوم تنها عدد 4 رو پاس بدید (هر آدرس IP از 4 بایت تشکیل میشه) و برای پارامتر آخر AF_INET. حالا شاید تعجب کنید که چرا ساختار جدید HOSTENT اینجا است. در واقع مقداری که تابع GetHostByAddr برمیگردونه به اشن ساختار اشاره داره اونجایی که پارامتر hName نام میزبان هست و بله، مقدارش هم Null هست. متاسفم ولی برای خارج کردن این ساختار و نام میزبان از مقدار بارگشتی باید از تابع RTLMoveMemory که با نام CopyMemory هم شناخته میشه کار کنید.

نتیجه گیری

خوب این یه مقاله مبتدی برای برنامه نویسی سطح پایین سوکت ها بود. امیدوارم لذت برده باشید و اگر هم از این مبحث خوشتون اومده میتونید برین سراغ یادگیری پروتوکل UDP و سوکت های Datagram و حتی بیشتر از اون سوکت های Raw. نقطه نظرات و پیشنهادات با کمال میل پذیرفته میشن. و باید بگم که من کد های این مقاله رو با دست طراحی کردم (heh!).

منابع و قدردانی ها

- دوست دارم تشکر مخصوصی از وب سایت www.vbip.com به جا بیارم به خاطر مقالات و نمونه برنامه های خوبش که برنامه نویسی اینترنت و شبکه و سوکت ها رو خیلی خوب دسته بندی کرده.
- تشکر بعدی نثار وب سایت www.AllApi.net میشه بخاطر لیست کامل و مرتبی که از API ها و مثال هاشون تهیه کردن و به موقع و خوب لیست API ها رو به روز رسانی هم میکنن.
- تشکر بعدی به سایت برنامه نویس تقدیم میشه که کمک های شایانی در شناخت و پیشرفت در رشته تحصیلی و کاری من شد.به امید موفقیت و سلامت روز افزور تمام مدیران و دوستان در سایت برنامه نویس.
- و در آخر، از شما دوستان تشکر میکنم که این وقت رو صرف کردید و این مقاله رو مطالعه کردین.