附录B　IP助手函数

本附录介绍一些新的API函数，有了这些函数，便可在自己的计算机上对 IP协议统计情况进行查询和管理。它们有助于获得下面的能力：

- Ipconfig.exe（或适用于微软Windows 95的Winipcfg.exe）：显示IP配置信息，允许释放和更新DHCP分配的IP地址。
- Netstat.exe：显示TCP连接表、UDP监听者表以及IP协议统计情况。
- Route.exe：显示并处理网络路由表。
- Arp.exe：显示并修改供“地址解析协议”（ARP）使用的IP到物理地址翻译表。

本附录介绍的这些函数主要用于 Windows 98和Windows 2000操作系统。有几个还可以用于Windows NT SP4及以后的SP（服务包）版本，但所有这些函数都不能用于 Windows 95。接下来的讨论中，我们将一一指出各个函数适用于哪些平台。本附录中的所有函数原型均定义在Iphlpapi,h文件中。在建立你自己的应用程序时，必须把它链接到这个库文件 Iphlpapi.lib。

B.1　IPCONFIG

Ipconfig.exe程序展示了两条信息：IP配置信息和IP配置参数，参数是由安装在机器上的网络适配器所决定的。要获得IP配置信息，利用GetNetworkParams函数即可。它的定义如下：

```
DWORD GetNetworkParams(
    PFIXED_INFO pFixedInfo,
    PULONG pOutBufLen
);
```

pFixedInfo参数取得一个缓冲区指针，该缓冲区接收 FIXED_INFO数据结构，你的应用程序必须提供这个结构，以便获得 IP配置信息。pOutBufLen参数是一个变量指针，指定投入pFixedInfo参数中的那个缓冲区的长度。如果你提供的缓冲区不够大，GetNetworkParams就会返回ERROR_BUFFER_OVERFLOW，并将pOutBufLen参数设为正确的缓冲区长度。GetNetworkParams中所用的FIXED_INFO结构的格式如下：

```
typedef struct
{
    char            HostName[MAX_HOSTNAME_LEN + 4] ;
    char            DomainName[MAX_DOMAIN_NAME_LEN + 4];
    PIP_ADDR_STRING CurrentDnsServer;
    IP_ADDR_STRING  DnsServerList;
    UINT            NodeType;
    char            ScopeId[MAX_SCOPE_ID_LEN + 4];
    UINT            EnableRouting;
    UINT            EnableProxy;
    UINT            EnableDns;
} FIXED_INFO, *PFIXED_INFO;
```

它的各个字段定义如下：

- HostName：代表你的计算机名，这个名字由DNS进行识别。

- DomainName：说明你的计算机属于哪个DNS域。
- CurrentDnsServer：包含当前DNS服务器的IP地址。
- DnsServerList：是一个链接列表，其中包含你的机器所采用的 DNS服务器。
- NodeType：说明IP网络上的系统是如何解析NetBIOS名的。表B-1包含了该字段可能的值
- ScopeId：识别一个字串值。这个值加在 NetBIOS名中，通过逻辑方式把两个或两个以上的计算机组在一起，在TCP/IP网络中进行通信。
- EnableRouting：说明系统是否会在它连接的网络中，路由 IP包。
- EnableProxy：说明系统是否充当网络上的 WINS代理。WINS代理通过WINS，响应它已解析过的名字查询，并允许由 b节点计算机组成的网络与其他已经用 WINS注册的子网上的服务器建立连接。
- EnableDns：说明NetBIOS是否向DNS查询WINS不能解析的名字、广播或 LMHOST文件。

表B-1　节点类型可能的值

值	说　明
BROADCAST_NODETYPE:	即b节点NetBIOS名字解析法，采用了这种解析方法，系统便利用IP广播来执行NetBIOS名字注册和名字解析
PEER_TO_PEER_NODETYPE:	即p节点名字解析，采用了这种解析方法，系统便利用点到点通信与一个NetBIOS名字服务器（比如WINS）通信，进而对IP地址进行注册和计算机名的解析
MIXED_NODETYPE:	即m节点（mixed节点）NetBIOS名字解析法，采用了这种解析法，系统便同时采用前面的b节点和p节点方法。先用b节点方法；如果失败，再用p节点方法
HYBRID_NODETYPE:	即h节点（hybrid节点）NetBIOS名字解析法，采用了这种解析法，系统便同时采用前面所讲的b节点和p节点。先用p节点；如果失败，再用b节点

FIXED_INFO结构的DnsServerList字段是一个IP_ADDR_STRING结构，代表IP地址链接列表的起始处。它的格式如下：

```
typedef struct _IP_ADDR_STRING
{
    struct _IP_ADDR_STRING* Next;
    IP_ADDRESS_STRING       IpAddress;
    IP_MASK_STRING          IpMask;
    DWORD                   Context;
} IP_ADDR_STRING, *PIP_ADDR_STRING;
```

Next字段标识列表中的下一个DNS服务器之IP地址。如果把它设为NULL，就表明列表中的最后一个地址。IpAddress字段是一个字符串，以点式十进制字串表示 Ip地址。IPMask字段也是一个字符串，表示子网掩码，这个掩码与 IpAddress中列出的IP地址关联在一起。最后一个字段是Context，用一个独一无二的值来标识系统上的 IP地址。

另外，利用IpConfig.exe程序，也可获得网络接口专有的 IP配置信息。网络接口不仅可以是一个硬件以太网适配器，甚至还可以是一个 RAS拨号适配器。调用 GetAdaptersInfo命令，便可获得适配器信息。该函数的定义如下：

```
DWORD GetAdaptersInfo (
    PIP_ADAPTER_INFO pAdapterInfo,
```

```
    PULONG pOutBufLen
);
```

通过pAdapterInfo参数，把一个指针投递给应用程序提供的缓冲区，这个缓冲区取得一个ADAPTER_INFO数据结构，该结构中含有这个适配器的配置信息。 pOutBufInfo参数是一个变量指针，这个变量指定投入pAdapterInfo参数中的缓冲区的长度。如果你提供的缓冲区不够大，GetAdaptersInfo就会返回ERROR_BUFFER_OVERFLOW，并把pOutBufLen参数设为所需要的缓冲区长度。

事实上，IP_ADAPTER_INFO结构是一个结构列表，你的机器上所有可用的网络适配器的IP配置信息都在这个列表中。IP_ADAPTER_INFO的格式如下：

```
typedef struct _IP_ADAPTER_INFO
{
    struct _IP_ADAPTER_INFO* Next;
    DWORD            ComboIndex;
    char             AdapterName[MAX_ADAPTER_NAME_LENGTH + 4];
    char             Description[MAX_ADAPTER_DESCRIPTION_LENGTH + 4];
    UINT             AddressLength;
    BYTE             Address[MAX_ADAPTER_ADDRESS_LENGTH];
    DWORD            Index;
    UINT             Type;
    UINT             DhcpEnabled;
    PIP_ADDR_STRING CurrentIpAddress;
    IP_ADDR_STRING   IpAddressList;
    IP_ADDR_STRING   GatewayList;
    IP_ADDR_STRING   DhcpServer;
    BOOL             HaveWins;
    IP_ADDR_STRING   PrimaryWinsServer;
    IP_ADDR_STRING   SecondaryWinsServer;
    time_t           LeaseObtained;
    time_t           LeaseExpires;
} IP_ADAPTER_INFO, *PIP_ADAPTER_INFO;
```

各字段定义如下：

- Next：缓冲区内的下一个适配器。 NULL值表明列表中的最后一个适配器。
- ComboIndex：未用，将设为0。
- AdapterName：适配器名。
- Description：是一个关于适配器的简单说明。
- AddressLength：Address字段中的那个适配器的物理地址由多少个字节组成。
- Address：该适配器的物理地址。
- Index：该适配器分配的唯一的网络接口内部索引编号。
- Type：将适配器类型指定为一个数字化的值。表 B-2定义了已获支持的适配器类型。
- DhcpEnabled：说明这个适配器上是否启用 DHCP。
- CurrentIpAddress：未用，但将设为NULL（空）值。
- IpAddressList：指定为这个适配器分配的 IP地址列表。
- GatewayList：指定代表默认网关的IP地址列表。
- DhcpServer：指定一个列表，表中只有一个元素，代表 DHCP服务器S所用的IP地址。
- HaveWins：说明该适配器是否使用 WINS服务器。

- PrimaryWinsServer：指定一个列表，表中只有一个元素，代表主 WINS服务器所用的IP地址。
- SecondaryWinsServer：指定一个列表，表中只有一个元素，代表辅助 WINS服务器所用的IP地址。
- LeaseObtained：标识何时开始租用DHCP服务器提供的IP地址。
- LeaseExpires：标识DHCP服务器提供的IP地址何时期满。

表B-2 适配器类型

适配器类型的值	说 明
MIB_IF_TYPE_ETHERNET	以太网适配器
MIB_IF_TYPE_FDDI	FDDI（光纤分布数据接口）适配器
MIB_IF_TYPE_LOOPBACK	Loopback适配器
MIB_IF_TYPE_OTHER	其他类型的适配器
MIB_IF_TYPE_PPP	PPP（点到点协议）适配器
MIB_IF_TYPE_SLIP	Slip适配器
MIB_IF_TYPE_TOKENRING	令牌环适配器

B.1.1 释放和更新IP地址

Ipconfig.exe程序的另一个特色是：能够释放和更新从 DCHP服务器处获得的IP地址。这是通过指定 /release和/renew命令行参数来完成的。如果想通过编程释放 IP地址，则可调用IPReleaseAddress函数，该函数定义如下：

```
DWORD IpReleaseAddress (
    PIP_ADAPTER_INDEX_MAP AdapterInfo
);
```

如果打算更新IP地址，则调用IPRenewAddress函数，它的定义如下：

```
DWORD IpRenewAddress (
    PIP_ADAPTER_INDEX_MAP AdapterInfo
);
```

这两个函数均突出了AdapterInfo参数是一个IP_ADAPTER_INDEX_MAP结构，该结构识别即将对其地址进行释放和更新的适配器。 IP_ADAPTER_INDEX_MAP结构的格式如下：

```
typedef struct _IP_ADAPTER_INDEX_MAP
{
    ULONG Index;
    WCHAR Name[MAX_ADAPTER_NAME];
}IP_ADAPTER_INDEX_MAP, *PIP_ADAPTER_INDEX_MAP;
```

它的各个字段是这样定义的：

- Index：标识为适配器分配的内部网络接口索引。
- Name：标识适配器名。

调用GetInterfaceInfo函数，便可获得某一特定适配器的IP_ADAPTER_INDEX_MAP结构。该函数的定义如下：

```
DWORD GetInterfaceInfo (
    IN PIP_INTERFACE_INFO pIfTable,
    OUT PULONG dwOutBufLen
);
```

pIfTable参数是一个指针，指向一个IP_INTERFACE_INFO应用程序缓冲区，该缓冲区将接收接口信息。dwOutBufLen参数是一个变量指针，这个变量指定投入pIfTable参数中的缓冲区的长度。如果这个缓冲区内容纳不下这个接口信息，GetInterfaceInfo便返回ERROR_INSUFFICIENT_BUFFER错误，并把dwOutBufLen参数设为合适的缓冲区长度。

IP_INTERFACE_INFO结构的格式如下；

```
typedef struct _IP_INTERFACE_INFO
{
    LONG                  NumAdapters;
    IP_ADAPTER_INDEX_MAP Adapter[1];
} IP_INTERFACE_INFO,*PIP_INTERFACE_INFO;
```

它的各个字段是这样定义的：

- NumAdapters：Adapter字段中的适配器编号。
- Adapter：是一个IP_ADAPTER_INDEX_MAP结构（见前面的定义）的数组。

一旦获得了特定适配器的IP_ADAPTER_INDEX_MAP结构，便可利用前面的IPReleaseAddress和IPRenewAddress这两个函数，释放或更新DHCP分配的IP地址了。

B.1.2 改变IP地址

Ipconfig.exe程序不允许改变网络适配器的IP地址（DHCP分配的除外）。但是，有两个函数允许你增添或删除特定适配器的IP地址，它们是：AddIpAddress和DeleteIpAddress IP助手函数。使用这两个函数时，需要知道网络适配器的索引编号和IP场景编号。Windows中，每个网络适配器都有一个独一无二的索引ID（前面已讲过），而且，每个IP地址都有一个独一无二的场景ID。适配器索引ID和IP场景编号都可通过GetAdaptersInfo获得。AddIpAddress函数的定义如下：

```
DWORD AddIPAddress (
    IPAddr Address,
    IPMask IpMask,
    DWORD IfIndex,
    PULONG NTEContext,
    PULONG NTEInstance
);
```

Address参数把准备增添的IP地址指定为一个无符号的长整数值。IpMask参数把IP地址的子网掩码指定为一个无符号的长整数值。IfIndex参数指定准备增添地址的适配器索引。NTEContext参数取得与所增添的IP地址关联的场景值。NTEInstance参数取得与一个IP地址关联的实例值。

如果想通过编程删除适配器的IP地址，调用DeleteIpAddress函数即可。它的定义如下：

```
DWORD DeleteIPAddress (
    ULONG NTEContext
);
```

NTEContext参数是与IP地址关联的值，这个值可从GetAdaptersInfo（前面已介绍）中得到。

B.2 NETSTAT

Netstat.exe程序显示了你的计算机上的TCP连接表、UDP监听者表以及IP协议统计。用于

获得这一信息的函数不仅可用于Windows 98和Windows 2000，而且可用于Windows NT 4 SP4（以及稍后的版本）。

B.2.1 取得TCP连接表

利用GetTcpTable函数，可获得TCP连接表。获得的信息和你在执行带上 -p tcp-a选项的Netstat.exe程序时看到的信息一样。GetTcpTable函数的定义如下：

```
DWORD GetTcpTable(
    PMIB_TCPTABLE pTcpTable,
    PDWORD pdwSize,
    BOOL bOrder
);
```

pTcpTable参数是一个指针，指向一个MIB_TCPTABLE应用程序缓冲区，该缓冲区内接收TCP连接信息。pdwsize参数是一个变量指针，这个变量指定投入 pTcpTable参数中的缓冲区长度。如果你提供的缓冲区容纳不下这个 TCP信息，函数就会把这个参数设为合适的缓冲区长度。bOrder参数指定是否对返回信息进行分类。

GetTcpTable函数返回的MIB_TCPTABLE结构的格式如下：

```
typedef struct _MIB_TCPTABLE
{
    DWORD dwNumEntries;
    MIB_TCPROW table[ANY_SIZE];
} MIB_TCPTABLE, *PMIB_TCPTABLE;
```

它的各个字段定义如下：

- dwNumEntries：说明table字段中有多少条目。
- table：是一个MIB_TCPROW结构数组，其中包含TCP连接信息。

MIB_TCPROW结构中包含构成一个TCP连接的IP地址对。它的格式如下：

```
typedef struct _MIB_TCPROW
{
    DWORD dwState;
    DWORD dwLocalAddr;
    DWORD dwLocalPort;
    DWORD dwRemoteAddr;
    DWORD dwRemotePort;
} MIB_TCPROW, *PMIB_TCPROW;
```

它的各个字段是这样定义的；

- dwState：指定TCP连接的状态，参见表B-3中的说明。

表B-3 TCP连接的状态

连接状态	RFC说明
MIB_TCP_STATE_CLOSED	“关闭”状态
MIB_TCP_STATE_CLOSING	“正在关闭”状态
MIB_TCP_STATE_CLOSE_WAIT	“关闭等待”状态
MIB_TCP_STATE_DELETE_TCB	“删除”状态
MIB_TCP_STATE_ESTAB	“已建立”状态
MIB_TCP_STATE_FIN_WAIT1	“FIN WAIT1”状态
MIB_TCP_STATE_FIN_WAIT2	“FIN WAIT2”状态

(续)

连接状态	RFC说明
MIB_TCP_STATE_LAST_ACK	“最后一次确认”状态
MIB_TCP_STATE_LISTEN	“正在监听”状态
MIB_TCP_STATE_SYN_RCVD	“同步接收”状态
MIB_TCP_STATE_SYN_SENT	“同步发送”状态
MIB_TCP_STATE_TIME_WAIT	“时间等待”状态

- dwLocalAddr：为连接指定一个本地IP地址。
- dwLocalPort：为连接指定一个本地端口。
- dwRemoteAddr：为连接指定远程IP地址。
- dwRemotePort：为连接指定远程端口。

B.2.2 取得UDP监听者表

利用GetUdpTable函数，可获得UDP监听者表。获得的信息和你在执行有一个 -pudp -a选项的Netstat.exe程序时看到的信息一样。GetUdpTable函数的定义如下：

```
DWORD GetUdpTable(
    PMIB_UDPTABLE pUdpTable,
    PDWORD pdwSize,
    BOOL bOrder
);
```

pUdpTable参数是一个指针，指向 MIB_UDPTABLE应用程序缓冲区，该缓冲区将接收UDP监听者信息。pdwSize参数是一个变量指针，这个变量指定投入 pUdpTable参数内的缓冲区的长度。如果缓冲区容纳不下这个 UDP信息，函数便把这个参数设为合适的缓冲区长度。bOrder参数指定是否对返回信息进行分类。

GetUdpTable函数返回的MIB_UDPTABLE结构格式如下：

```
typedef struct _MIB_UDPTABLE
{
    DWORD dwNumEntries;
    MIB_UDPROW table[ANY_SIZE];
} MIB_UDPTABLE, * PMIB_UDPTABLE;
```

它的各个字段是这样定义的：

- dwNumEntries：说明table字段中有多少条目。
- table：是一个MIB_UDPROW结构数组，其中包含UDP监听者的信息。

MIB_UDPROW结构中包含IP地址，UDP正在这些地址上监听数据报。它的格式如下：

```
typedef struct _MIB_UDPROW
{
    DWORD dwLocalAddr;
    DWORD dwLocalPort;
} MIB_UDPROW, * PMIB_UDPROW;
```

该结构的各个字段是这样定义的：

- dwLocalAddr：指定本地IP地址。
- dwLocalPort：指定本地IP端口。

B.2.3 取得IP协议统计情况

用于获得IP统计情况的四个函数是：GetIpStatistics、GetIcmpStatistics、GetTcpStatistics以及GetUdpStatistics。这些函数产生的信息和调用带上 -s参数的Netstat.exe程序时返回的信息一样。利用第一个统计函数 GetIpStatistics，可获得当前计算机上的 IP统计情况，它的定义如下：

```
DWORD GetIpStatistics(
    PMIB_IPSTATS pStats
);
```

pStats参数是一个指针，指向MIB_IPSTATA结构，这个结构接收你的计算机当前的 IP统计情况。MIB_IPSTATS结构的格式如下：

```
typedef struct _MIB_IPSTATS
{
    DWORD dwForwarding;
    DWORD dwDefaultTTL;
    DWORD dwInReceives;
    DWORD dwInHdrErrors;
    DWORD dwInAddrErrors;
    DWORD dwForwDatagrams;
    DWORD dwInUnknownProtos;
    DWORD dwInDiscards;
    DWORD dwInDelivers;
    DWORD dwOutRequests;
    DWORD dwRoutingDiscards;
    DWORD dwOutDiscards;
    DWORD dwOutNoRoutes;
    DWORD dwReasmTimeout;
    DWORD dwReasmReqds;
    DWORD dwReasmOks;
    DWORD dwReasmFails;
    DWORD dwFragOks;
    DWORD dwFragFails;
    DWORD dwFragCreates;
    DWORD dwNumIf;
    DWORD dwNumAddr;
    DWORD dwNumRoutes;
} MIB_IPSTATS, *PMIB_IPSTATS;
```

该结构的各个字段是这样定义的：

- dwForwarding：说明你的计算机上是启用，还是禁止转发 IP包。
- dwDefaultTTL：为你的计算机所发出的数据报指定最初的生存期（ TTL）的值。
- dwInReceives：说明已收到多少数据报。
- dwInHdrErrors：说明已收到多少报头有误的数据报。
- dwInAddrErrors：说明已收到多少地址有误的数据报。
- dwForwDatagrams：说明已转发多少数据报。
- dwInUnknownProtos：说明已收到多少协议不明的数据报。
- dwInDiscards：说明已收到多少已丢弃的数据报。
- dwInDelivers：说明已收到多少已投递的数据报。

- dwOutRequests：说明IP请求传输多少数据报。
- dwRoutingDiscards：说明已丢弃的外出数据报有多少。
- dwOutDiscards：说明丢弃的传输数据报有多少。
- dwOutNoRoutes：说明没有路由目标的数据报有多少。
- dwReasmTimeout：说明分段数据报完全到达的最长时间。
- dwReasmReqds：说明需要重组的数据报有多少。
- dwReasmOks：说明已成功重组的数据报有多少。
- dwFragFails：说明不能进行分段的数据报有多少。
- dwFragCreates：说明可被分段的数据报有多少。
- dwNumIf：说明你的计算机上可用的IP接口有多少。
- dwNumAddr：说明你的计算机上标识的IP地址有多少。
- dwNumRoutes：说明路由表中可用的路由有多少。

第二个统计函数：GetIcmpStatistics，用于获得“互联网控制协议”（ICMP）统计情况。它的定义如下：

```
DWORD GetIcmpStatistics(
   PMIB_ICMP pStats
);
```

pStats参数是一个指针，指向MIB_ICMP结构，这个结构接收你的计算机上当前的ICMP统计情况。MIB_ICMP结构的格式如下：

```
typedef struct _MIB_ICMP
{
    MIBICMPINFO stats;
} MIB_ICMP,*PMIB_ICMP;
```

一眼可见，这个MIB_ICMP结构中另包含一个MIBICMPINFO结构，后者的格式如下：

```
typedef struct _MIBICMPINFO
{
    MIBICMPSTATS icmpInStats;
    MIBICMPSTATS icmpOutStats;
} MIBICMPINFO;
```

MIBICMPINFO结构通过MIBICMPSTATS结构接收接入或外出的ICMP信息。icmpInStats参数接收接入数据，而icmpOutStats参数则接收外出数据。MIBICMPSTATS结构的格式如下：

```
typedef struct _MIBICMPSTATS
{
    DWORD dwMsgs;
    DWORD dwErrors;
    DWORD dwDestUnreachs;
    DWORD dwTimeExcds;
    DWORD dwParmProbs;
    DWORD dwSrcQuenchs;
    DWORD dwRedirects;
    DWORD dwEchos;
    DWORD dwEchoReps;
    DWORD dwTimestamps;
    DWORD dwTimestampReps;
    DWORD dwAddrMasks;
    DWORD dwAddrMaskReps;
```

```
} MIBICMPSTATS;
```

它的各个字段是这样定义的：

- dwMsgs：说明已收发多少消息。
- dwErrors：说明已收发多少错误。
- dwDestUnreachs：说明已收发多少“目标不可抵达”消息。
- dwTimeExcds：说明已收发多少生存期已过消息。
- dwParmProbs：说明已收发多少表明数据报内有错误IP信息的消息。
- dwSrcQuenchs：说明已收发多少源结束消息。
- dwRedirects：说明已收发多少重定向消息。
- dwEchos：说明已收发多少ICMP响应请求。
- dwEchoReps：说明已收发多少ICMP响应应答。
- dwTimestamps：说明已收发多少时间戳请求。
- dwTimestampReps：说明已收发多少时间戳响应。
- dwAddrMasks：说明已收发多少地址掩码。
- dwAddrMaskReps：说明已收发多少地址掩码响应。

用于获得TCP统计情况的第三个统计函数是GetTcpStatistcs，它的定义如下：

```
DWORD GetTcpStatistics(
    PMIB_TCPSTATS pStats
);
```

pStats参数是一个指针，指向MIB_TCPSTATS结构，这个结构接收你的计算机当前的IP统计。MIB_TCPSTATS结构的格式如下：

```
typedef struct _MIB_TCPSTATS
{
    DWORD dwRtoAlgorithm;
    DWORD dwRtoMin;
    DWORD dwRtoMax;
    DWORD dwMaxConn;
    DWORD dwActiveOpens;
    DWORD dwPassiveOpens;
    DWORD dwAttemptFails;
    DWORD dwEstabResets;
    DWORD dwCurrEstab;
    DWORD dwInSegs;
    DWORD dwOutSegs;
    DWORD dwRetransSegs;
    DWORD dwInErrs;
    DWORD dwOutRsts;
    DWORD dwNumConns;
} MIB_TCPSTATS, *PMIB_TCPSTATS;
```

它的各个字段是这样定义的：

- dwRtoAlgorithm：说明即将采用哪种重传输算法。有效值包括： MIB_TCP_RTO_CONSTANT、MIB_TCP_RTO_RSRE、MIB_TCP_RTO_VANJ以及针对其他类型的MIB_TCP_RTO_OTHER。
- dwRtoMin：说明重传输超时的最小值，以毫秒计。
- dwRtoMax：说明重传输超时的最大值，以毫秒计。

- dwMaxConn：说明最多能接受多少连接。
- dwActiveOpens：说明你的计算机向服务器发起了多少次连接。
- dwPassiveOpens：说明你的计算机监听了多少次客户机发出的连接。
- dwAttemptFails：说明尝试连接失败的次数是多少。
- dwEstabResets：说明对已建立的连接实行了多少次重设。
- dwCurrEstab：说明目前已建立的连接有多少。
- dwInSegs：说明收到了多少分段数据报。
- dwOutSegs：说明传输了多少分段数据报（除已重新传输的分段外）。
- dwRetransSegs：说明已传输了多少分段数据报。
- dwInErrs：说明收到多少错误。
- dwOutRsts：说明重设标志后，又传输了多少分段数据报。
- dwNumConns：说明连接的总数是多少。

最后一个统计函数GetUdpStatistics，用于获得你的计算机上的UDP统计情况。它的定义如下：

```
DWORD GetUdpStatistics(
    PMIB_UDPSTATS pStats
);
```

pStats参数是一个指针，指向MIB_UDPSTATS结构，这个结构接收你的计算机当前的IP统计。MIB_UDPSTATS结构的格式如下：

```
typedef struct _MIB_UDPSTATS
{
    DWORD dwInDatagrams;
    DWORD dwNoPorts;
    DWORD dwInErrors;
    DWORD dwOutDatagrams;
    DWORD dwNumAddrs;
} MIB_UDPSTATS,*PMIB_UDPSTATS;
```

它的各个字段是这样定义的：

- dwInDatagrams：说明已收到多少数据报。
- dwNoPorts：说明因为端口号有误而丢弃了多少数据报。
- dwInErrors：说明已收到多少错误数据报（除dwNoPorts中统计的数目之外）。
- dwOutDatagrams：说明已传输多少数据报。
- dwNumAddrs：说明监听者表中有多少UDP条目。

B.3 ROUTE

利用Route.exe命令，我们可以打印和修改路由表。路由表用于决定连接请求或收发数据报在哪个IP接口上进行。“IP助手库”（IP Helper Library）提供了几个函数，用于对路由表进行处理。这几个函数可用于Windows 98，Windows 2000以及Windows NT 4 SP4（或以后的版本）。

首先，为大家讲讲Route.exe命令的作用。该命令最基本的用法便是打印路由表。一个路由的组成有这几个要素：一个目标地址、一个网络掩码、一个网关、一个本地 IP接口和一个

公制。另外的用法便是增添和删除路由。若想增添路由，必须指定前面提到的全部参数。若想删除路由，只指定目标地址即可。在本小节，我们准备看看这几个 IP助手函数是怎样打印路由表的。随后，再为大家讲讲怎样增添或删除路由。

B.3.1 获得路由表

Route.exe执行的最常见操作便是打印路由表。这是通过 GetIpForwardTable函数来完成的。该函数的定义如下：

```
DWORD GetIpForwardTable (
    PMIB_IPFORWARDTABLE pIpForwardTable,
    PULONG pdwSize,
    BOOL bOrder
);
```

第一个参数pIpForwardTable中包含了函数返回的路由表信息。在调用这个函数时，该参数应该引用一个恰当的缓冲区。如果 pIpForwardTable参数被设为NULL（或者缓冲区长度不够），pdwSize参数便返回成功调用该函数所需的缓冲区的长度。最后一个参数 bOrder，表明是否对返回结果进行分类。默认的分类顺序是：

1) 目标地址。
2) 生成路由的协议。
3) 多路径路由策略。
4) 下一跳的地址。

路由信息包含在MIB_INFORWARDROW结构中，这个结构的格式如下：

```
typedef struct _MIB_IPFORWARDTABLE
{
    DWORD             dwNumEntries;
    MIB_IPFORWARDROW table[ANY_SIZE];
} MIB_IPFORWARDTABLE, *PMIB_IPFORWARDTABLE;
```

这个结构内还有一个MIB_IPFORWARDROW结构数组。dwNumEntries字段表明这个数组中有几个结构。MIB_IPFORWARDROW结构的格式如下：

```
typedef struct _MIB_IPFORWARDROW
{
    DWORD dwForwardDest;
    DWORD dwForwardMask;
    DWORD dwForwardPolicy;
    DWORD dwForwardNextHop;
    DWORD dwForwardIfIndex;
    DWORD dwForwardType;
    DWORD dwForwardProto;
    DWORD dwForwardAge;
    DWORD dwForwardNextHopAS;
    DWORD dwForwardMetric1;
    DWORD dwForwardMetric2;
    DWORD dwForwardMetric3;
    DWORD dwForwardMetric4;
    DWORD dwForwardMetric5;
} MIB_IPFORWARDROW, *PMIB_IPFORWARDROW;
```

它的各个字段是这样定义的：

- dwForwardDest：是目标主机的IP地址。
- dwForwardMask：是目标主机的子网掩码。
- dwForwardPolicy：指定影响多路径路由选择的一系列条件。这些条件通常采用“ IP的服务类型”（TOS）的形式。关于TOS的详情，可参考第9章和IP_TOS选项。关于多路径路由的详情，参考1354。
- dwForwardNextHop：是路由表中下一跳的IP地址。
- dwForwardIfIndex：表明针对该路由的接口之索引。
- dwForwardType：表明RFC 1354中定义的路由类型。表B-4列出了该字段可能的值及其含义。
- dwForwardProto：是生成这个路由的协议。表 B-5列出了该字段可能的值。 IPX协议值定义在Routprot.h中，而IP条目则包含在Iprtrmib.h中。
- dwForwardAge：表明路由持续时间，以毫秒计。
- dwForwardNextHopAS：是下一跳的自治系统编号。
- dwForwardMetric1：是路由协议专有的公制值。详情参见RFC 1354。这个字段中包含路由公制值，这个值是在执行Route.exe打印命令时经常看到的。若不使用这个条目，对这个字段以及下面四个字段而言，其值均为MIB_IPROUTE_METRIC_UNUSED（-1）。
- dwForwardMetric2：是路由协议专有的公制值。详情参见 RFC 1354。
- dwForwardMetric3：是路由协议专有的公制值。详情参见 RFC 1354。
- dwForwardMetric4：是路由协议专有的公制值。详情参见 RFC 1354。
- dwForwardMetric5：是路由协议专有的公制值。详情参见 RFC 1354。

表B-4 路由表条目中可能的路由类型

转发类型	说 明
MIB_IPROUTE_TYPE_INDIRECT	下一跳不是最终目的地（远程路由）
MIB_IPROUTE_TYPE_DIRECT	下一跳是最终目的地（本地路由）
MIB_IPROUTE_TYPE_INVALID	路由无效
MIB_IPROUTE_TYPE_OTHER	其他路由

表B-5 路由协议标识符

协议标识符	说 明
MIB_IPPROTO_OTHER	未列出的协议
MIB_IPPROTO_LOCAL	堆栈生成的路由
MIB_IPPROTO_NETMGMT	Route.exe程序或SNMP添加的路由器
MIB_IPPROTO_ICMP	ICMP重定向的路由
MIB_IPPROTO_EGP	外部网关协议
MIB_IPPROTO_GGP	网关网关协议
MIB_IPPROTO_HELLO	HELLO路由协议
MIB_IPPROTO_RIP	路由信息协议
MIB_IPPROTO_IS_IS	IP中间系统到中间系统协议
MIB_IPPROTO_ES_IS	IP终端系统到中间系统协议
MIB_IPPROTO_CISCO	IPCisco协议
MIB_IPPROTO_BBN	BBN协议
MIB_IPPROTO_OSPF	先打开最短路径（OSPF）路由协议
MIB_IPPROTO_BGP	边.界网关协议

(续)

协议标识符	说 明
MIB_IPPROTO_NT_AUTOSTATIC	最初由一个路由协议添加的、非静态路由
MIB_IPPROTO_NT_STATIC	路由用户接口或Routemon.exe程序添加的路由
MIB_IPPROTO_STATIC_NON_DOD	等同于PROTO_IP_NT_STATIC，但这些路由不会引发“按需拨号”（DOD）
IPX_PROTOCOL_RIP	IPX的路由信息协议
IPX_PROTOCOL_SAP	服务声明协议
IPX_PROTOCOL_NLSP	NetWare链接服务协议

B.3.2 增加路由

路由命令的另一个作用是添加路由。记住，要添加一个路由，必须指定其目标 IP、网络掩码、网关、本地IP接口以及公制。添加路由时，应该保证所给的本地 IP接口是有效的。除此以外，还需要根据本地IP接口的内部索引值，引用这个接口，因为路由将添加到这个接口上。调用GetIpAddrTable函数，便可获得这一信息。该函数的定义如下：

```
DWORD GetIpAddrTable (
    PMIB_IPADDRTABLE pIpAddrTable,
    PULONG pdwSize,
    BOOL bOrder
);
```

第一个参数pIpAddrTable，是将返回MIB_IPADDRTABLE结构的缓冲区的正确长度。pdwSize参数是被当作第一个参数投递的缓冲区的长度。最后一个参数 bOrder，说明是否返回以升序的方式处理IP地址的本地IP接口。要找到正确的缓冲区长度，可指定 pIpAddrTable参数为NULL。函数返回之后，pdwSize便指定正确的缓冲区长度。MIB_IPADDRTABLE结构的格式如下：

```
typedef struct _MIB_IPADDRTABLE
{
    DWORD dwNumEntries
    MIB_IPADDRROW table[ANY_SIZE];
} MIB_IPADDRTABLE, *PMIB_IPADDRTABLE;
```

这个结构内还有一个MIB_IPADDROW结构。dwNumEntries字段表明table字段数组中有多少MIB_IPADDROW条目。MIB_IPADDRROW结构的格式如下：

```
typedef struct _MIB_IPADDRROW
{
    DWORD    dwAddr;
    DWORD    dwIndex;
    DWORD    dwMask;
    DWORD    dwBCastAddr;
    DWORD    dwReasmSize;
    unsigned short unused1;
    unsigned short unused2;
} MIB_IPADDRROW, *PMIB_IPADDRROW;
```

它的各个字段是这样定义的：

- dwAddr：是指定接口的IP地址。
- dwIndex：与IP地址关联在一起的接口之索引。

- dwMask：是IP地址的子网掩码。
- dwBCastAddr：是广播地址。
- dwReasmSize：是已收到的数据报重装后的最大长度。
- unused1 and unused2：目前尚未使用。

利用这个函数，便可验证针对指定路由的本地 IP地址是否有效。在路由表中添加路由的函数是SetIpForwardEntry，它的定义如下：

```
DWORD SetIpForwardEntry (
    PMIB_IPFORWARDROW pRoute
);
```

该函数中，唯一的参数pRoute，是一个指向MIB_IPFORWARDROW结构的指针。这个结构定义了建立一个新路由所需要的元素。我们曾对这个结构及其成员字段进行了讨论。要添加一个新路由，必须指定下面几个字段的值：dwForwardIfIndex、dwForwardDest、dwForwardMask、dwForwardNextHop以及dwForwardPolicy。

B.3.3 删除路由

路由程序的最后一个行动是删除路由，这个命令最简单。调用路由命令删除路由时，必须指定准备删除的目标地址。然后，才能搜索与目标地址对应的 MIB_IPFORWARDROW结构，这个结构是 GetIpForwardTable返回的。接下来，再将 MIB_IPFORWARDROW结构投递给DeleteForwardEntry函数，便可删除指定的路由条目了。 DeleteForwardEntry函数的定义如下：

```
DWORD DeleteIpForwardEntry (
    PMIB_IPFORWARDROW pRoute
);
```

删除路由的另一种可选办法是自行指定pRo，ute字段。删除路由所需的字段有：dwForward-IfIndex、dwForwardDest、dwForwardMask、dwForwardNextHop以及dwForwardPolicy。

B.4 ARP

Arp.exe程序用于查看和处理 ARP缓存。通过IP助手函数仿真 Arp.exe程序的Platform SDK范例名为Iprap.exe。ARP（也许大家还记得，它代表名字解析协议）负责把一个 IP地址解析成一个物理性的MAC地址。为不致影响性能，计算机将这一信息缓存下来，并可能通过 Arp.exe程序对该信息进行访问。利用这个程序，选择- a选项，便显示ARP表；选择- d选项，便删除一个条目；选择- s选项，则添加一个条目。下一小节中，我们将谈谈如何打印 ARP缓存，在ARP表中添加条目以及删除ARP条目。

本小节中讨论的 IP助手函数适用于 Windows 98、Windows 2000以及Windows NT4 SP4（以及稍后的版本）。

用于获得ARP表的这个IP助手函数最简单。它便是GetIpNetTable，其定义如下：

```
DWORD GetIpNetTable (
    PMIB_IPNETTABLE pIpNetTable,
    PULONG          pdwSize,
    BOOL            bOrder
);
```

第一个参数pIpNetTable，是一个指向MIB_IPNETTABLE结构的指针，这个结构返回的是ARP

信息。在调用这个函数时，必须提供一个足够大的缓冲区长度。和其他IP助手函数一样，如该参数是NULL，就会返回pdwSize参数所需的正确的缓冲区长度和ERROR_INSUFFICIENT_BUFFER错误。反之，pdwSize则表明被当作pIpNetTable投递的缓冲区的长度。最后一个参数 bOrder，表明是否按升序对返回的IP条目进行分类。

MIB_IPNETTABLE结构内还有一个 MIB_IPNETROW结构数组，后者的格式如下：

```
typedef struct _MIB_IPNETTABLE
{
    DWORD        dwNumEntries;
    MIB_IPNETROW table[ANY_SIZE];
} MIB_IPNETTABLE, *PMIB_IPNETTABLE;
```

dwNumEntries字段表明table字段中有多少数组条目。 MIB_IPNETROW结构中包含真正的ARP条目信息。它的格式如下：

```
typedef struct _MIB_IPNETROW {
    DWORD dwIndex;
    DWORD dwPhysAddrLen;
    BYTE  bPhysAddr[MAXLEN_PHYSADDR];
    DWORD dwAddr;
    DWORD dwType;
} MIB_IPNETROW, *PMIB_IPNETROW;
```

该结构的各个字段定义如下：

- dwIndex：指定适配器的索引
- dwPhysAddrLen：表明bPhysAddrs字段内包含的物理接口的长度，按字节数计
- bPhysAddr：是一个字节数组，其中包含适配器的物理地址
- dwAddr：指定适配器的IP地址
- dwType：表明ARP条目的类型。表B-6展示了这个字段可能的值

表B-6 ARP条目类型可能的值

ARP类型	含 义
MIB_IPNET_TYPE_STATIC	静态条目
MIB_IPNET_TYPE_DYNAMIC	动态条目
MIB_IPNET_TYPE_INVALID	无效条目
MIB_IPNET_TYPE_OTHER	其他条目

B.4.1 添加ARP条目

ARP另一个作用是在 ARP缓存中添加条目，这个操作比较简单。用于添加 ARP条目的IP助手函数是SetIpNetEntry，它的定义如下：

```
DWORD SetIpNetEntry (
    PMIB_IPNETROW pArpEntry
);
```

唯一的参数是MIB_IPNETWORK结构，我们已在前一小节中提过这个结构。要添加一个ARP条目，只须用新ARP信息来填充这个结构。首先，把 dwIndex参数设为一个本地IP地址的索引，这个索引表明添加的 ARP条目将用于哪个网络。记住，如果给出的是 IP地址，就可利

用GetIpAddrTable函数把它映射到索引。下一个字段 dwPhysAddrLen，一般设为6（多数物理地址，比如说ETHEENET MAC地址，长度都是6个字节）。bPhysAddr字节数组必须设为物理地址。多数MAC地址都用12个字符来表示，比如00-A0-C9-86-E8。这些字符需要被硬编码到bPhysAddr字段的恰当的字节数组位置内。比如， MAC地址范例就会硬编码到下面的字节中：

```
00000000 10100000 11001001 10100111 10000110 11101000
```

编码方法和编码IPX及ATM地址所采用的方法一样（详情参见第 6章）。dwAddr字段必须设为远程主机的IP地址，此时正在该机器上输入它的 MAC地址。最后一个字段dwType，设为表B-6中所列的ARP条目类型之一。MIB_IPNETWORK结构填充好之后，就调用 SetIpNetEntry函数，在缓存中添加ARP条目。如果成功，就会返回NO_ERROR。

B.4.2 删除ARP条目

删除ARP条目类似于添加条目，只不过删除时需要的信息有所不同，这里需要的是接口索引dwIndex，和准备删除的 ARP条目之IP地址 —— dwADDR。用于删除 ARP条目的函数是DeleteIpNetEntry，它的定义如下：

```
DWORD DeleteIpNetEntry (
    PMIB_IPNETROW pArpEntry
);
```

其唯一的参数是一个MIB_IPNETROW结构，删除ARP条目时所需的唯一的信息是本地 IP索引和准备删除的IP地址。记住，本地IP接口的索引编号可通过GetIpAddrTable函数获得。如果成功，便返回NO_ERROR。