当 Socket 与 TCP/IP 合用时，Socket 就是 TCP/IP 的通讯端点。因此，Socket 指定了 IP 位址和埠号。

网路时间服务

下面给出的范例程式与提供时间协定（Time Protocol）的 Internet 伺服器相连结。此程式将获得目前准确的日期和时间，并用此资讯设定您的 PC 时钟。

在美国，国家标准和技术协会（National Institute of Standards and Technology）（以前称为国家标准局（National Bureau of Standards））负责维护准确时间，该时间与世界各地的机构相联系。准确时间可用於无线电广播、电话号码、电脑拨号电话号码以及 Internet，关於这些的所有文件都位於网站 http://www.bldrdoc.gov/timefreq （网域名称「bldrdoc」指的是 Boulder、Colorado、NIST Time 的位置和 Frequency Division）。

我们只对 NIST Network Time Service 感兴趣，其详细的文件位於 http://www.bldrdoc.gov/timefreq/service/nts.htm 。此网页列出了十个提供 NIST 时间服务的伺服器。例如，第一个名称为 time-a.timefreq.bldrdoc.gov，其 IP 位址为 132.163.135.130。

（我曾经编写过一个使用非 Internet NIST 电脑拨接服务的程式，并发表於《PC Magazine》，您也可以在 Ziff-Davis 的网站 http://www.zdnet.com/pcmag/pctech/content/16/20/ut1620.001.html 中找到。此程式对於想学习如何使用 Windows Telephony API 的人很有帮助。）

在 Internet 上有三个不同的时间服务，每一个都由 Request for Comment（RFC）描述为 Internet 标准。日期协定（Daytime Protocol）（RFC-867）提供了一个 ASCII 字串用於指出准确的日期和时间。该 ASCII 字串的准确格式并不标准，但人们可以理解其中的含义。时间协定（RFC-868）提供了一个 32 位元的数字，用来表示从 1900 年 1 月 1 日至今的秒数。该时间是 UTC（不考虑字母顺序，它表示世界时间座标（Coordinated Universal Time）），它类似於所谓的格林威治标准时间（Greenwich Mean Time）或者 GMT——英国格林威治时间。第三个协定称为网路时间协定（Network Time Protocol）（RFC-1305），该协定很复杂。

对於我们的目的，即包括分析 Socket 和不断更新 PC 时钟，时间协定 RFC-868 已经够用了。RFC-868 只是一个两页的简短文件，主要是说用 TCP 获得准确时间的程式应该有如下步骤：

1. 连结到提供此服务的伺服器埠 37。
2. 接收 32 位元的时间。

3. 关闭连结。

4. 现在我们已经知道了编写存取时间服务的 Socket 应用程式的每个细节。

NETTIME 程式

Windows Sockets API，通常也称为 WinSock，与 Berkeley Sockets API 相容，因此，可以想像 UNIX Socket 程式码可以顺利地拿到 Windows 上使用。Windows 下更进一步的支援由对 Berkeley Socket 扩充的功能提供，其函式的形式是以 WSA(「WinSock API」)为字首。相关的概述和参考位於/Platform SDK/Networking and Distributed Services/Windows Sockets Version 2。

NETTIME，如程式 23-1 所示，展示了使用 WinSock API 的方法。

程式 23-1 NETTIME

```
NETTIME.C
/*------------------------------------------------------------------------
--
      NETTIME.C --      Sets System Clock from Internet Services
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------
-*/

#include <windows.h>
#include "resource.h"

#define     WM_SOCKET_NOTIFY (WM_USER + 1)
#define     ID_TIMER        1

LRESULT     CALLBACK WndProc   (HWND, UINT, WPARAM, LPARAM) ;
BOOL              CALLBACK MainDlg   (HWND, UINT, WPARAM, LPARAM) ;
BOOL              CALLBACK ServerDlg           (HWND, UINT, WPARAM, LPARAM) ;

void  ChangeSystemTime       (HWND hwndEdit, ULONG ulTime) ;
void  FormatUpdatedTime      (HWND hwndEdit, SYSTEMTIME * pstOld,
                           SYSTEMTIME * pstNew) ;
void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...) ;
HINSTANCE hInst ;
HWND              hwndModeless ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                 PSTR szCmdLine, int iCmdShow)
{
      static TCHAR     szAppName[] = TEXT ("NetTime") ;
      HWND                          hwnd ;
      MSG                     msg ;
```

```
     RECT                        rect ;
     WNDCLASS                    wndclass ;

     hInst = hInstance ;
     wndclass.style                   = 0 ;
     wndclass.lpfnWndProc             = WndProc ;
     wndclass.cbClsExtra              = 0 ;
     wndclass.cbWndExtra              = 0 ;
     wndclass.hInstance               = hInstance ;
     wndclass.hIcon                   = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor                 = NULL ;
     wndclass.hbrBackground      = NULL ;
     wndclass.lpszMenuName       = NULL ;
     wndclass.lpszClassName      = szAppName ;

     if (!RegisterClass (&wndclass))
     {
       MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                 szAppName, MB_ICONERROR) ;
                 return 0 ;
   }

     hwnd = CreateWindow (szAppName, TEXT ("Set System Clock from Internet"),
                     WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU |
                      WS_BORDER | WS_MINIMIZEBOX,
                                 CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

          // Create the modeless dialog box to go on top of the window
     hwndModeless = CreateDialog (hInstance, szAppName, hwnd, MainDlg) ;
          // Size the main parent window to the size of the dialog box.
          //   Show both windows.

     GetWindowRect (hwndModeless, &rect) ;
     AdjustWindowRect (&rect, WS_CAPTION | WS_BORDER, FALSE) ;

     SetWindowPos (   hwnd, NULL, 0, 0, rect.right - rect.left,
                     rect.bottom - rect.top, SWP_NOMOVE) ;

     ShowWindow (hwndModeless, SW_SHOW) ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     // Normal message loop when a modeless dialog box is used.
     while (GetMessage (&msg, NULL, 0, 0))
     {
                if (hwndModeless == 0 || !IsDialogMessage (hwndModeless, &msg))
```

```
                {
                        TranslateMessage (&msg) ;
                        DispatchMessage (&msg) ;
                }
      }
      return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
      switch (message)
     {
      case  WM_SETFOCUS:
                        SetFocus (hwndModeless) ;
                        return 0 ;

      case  WM_DESTROY:
                        PostQuitMessage (0) ;
                        return 0 ;
     }
      return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK MainDlg (      HWND hwnd, UINT message, WPARAM wParam,
                                                LPARAM lParam)
{
      static char             szIPAddr[32] = { "132.163.135.130" } ;
      static HWND             hwndButton, hwndEdit ;
      static SOCKET           sock ;
      static struct           sockaddr_in sa ;
      static TCHAR            szOKLabel[32] ;
      int                                     iError, iSize ;
      unsigned long           ulTime ;
      WORD                                    wEvent, wError ;
      WSADATA                           WSAData ;

      switch (message)
      {
      case  WM_INITDIALOG:
                        hwndButton = GetDlgItem (hwnd, IDOK) ;
                        hwndEdit = GetDlgItem (hwnd, IDC_TEXTOUT) ;
                        return TRUE ;

      case  WM_COMMAND:
                        switch (LOWORD (wParam))
                {
                        case  IDC_SERVER:
```

```
                        DialogBoxParam (hInst, TEXT ("Servers"), hwnd, ServerDlg,
(LPARAM) szIPAddr) ;
                        return TRUE ;

                        case  IDOK:
                  // Call "WSAStartup" and display description text

                        if (iError = WSAStartup (MAKEWORD(2,0), &WSAData))
                        {
                  EditPrintf (hwndEdit, TEXT ("Startup error #%i.\r\n"), iError) ;
                  return TRUE ;
                              }
                        EditPrintf (hwndEdit, TEXT ("Started up %hs\r\n"),
                            WSAData.szDescription);

                  // Call "socket"

                              sock = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP) ;

                                          if (sock == INVALID_SOCKET)
                                          {
                  EditPrintf (hwndEdit, TEXT ("Socket creation error #%i.\r\n"),
WSAGetLastError ()) ;
                WSACleanup () ;
                return TRUE ;
                   }
               EditPrintf (hwndEdit, TEXT ("Socket %i created.\r\n"), sock) ;

              // Call "WSAAsyncSelect"

              if (SOCKET_ERROR == WSAAsyncSelect (sock, hwnd, WM_SOCKET_NOTIFY,
FD_CONNECT | FD_READ))
                                          {
              EditPrintf (     hwndEdit, TEXT ("WSAAsyncSelect error #%i.\r\n"),
WSAGetLastError ()) ;
                closesocket (sock) ;
                WSACleanup () ;
                return TRUE ;
                   }

                // Call "connect" with IP address and time-server port

                   sa.sin_family         = AF_INET ;
                   sa.sin_port           = htons (IPPORT_TIMESERVER) ;
                   sa.sin_addr.S_un.S_addr = inet_addr (szIPAddr) ;

                   connect(sock, (SOCKADDR *) &sa, sizeof (sa)) ;
```

```
            // "connect" will return SOCKET_ERROR because even if it
            // succeeds, it will require blocking. The following only
            // reports unexpected errors.

            if (WSAEWOULDBLOCK != (iError = WSAGetLastError ()))
                 {
            EditPrintf (hwndEdit, TEXT ("Connect error #%i.\r\n"), iError) ;
                      closesocket (sock) ;
                      WSACleanup () ;
                      return TRUE ;
            }
            EditPrintf (hwndEdit, TEXT ("Connecting to %hs..."), szIPAddr) ;

            // The result of the "connect" call will be reported
            // through the WM_SOCKET_NOTIFY message.
            // Set timer and change the button to "Cancel"

           SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
           GetWindowText (hwndButton, szOKLabel, sizeof (szOKLabel) /sizeof
(TCHAR)) ;
           SetWindowText (hwndButton, TEXT ("Cancel")) ;
           SetWindowLong (hwndButton, GWL_ID, IDCANCEL) ;
          return TRUE ;

                      case  IDCANCEL:
                                     closesocket (sock) ;
                                     sock = 0 ;
                                     WSACleanup () ;
                                SetWindowText (hwndButton, szOKLabel) ;
                                SetWindowLong (hwndButton, GWL_ID, IDOK) ;

                                     KillTimer (hwnd, ID_TIMER) ;
                                     EditPrintf      (hwndEdit,      TEXT
("\r\nSocket closed.\r\n")) ;
                                     return TRUE ;

                      case  IDC_CLOSE:
                                 if (sock)
                      SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;

                                     DestroyWindow (GetParent (hwnd)) ;
                                     return TRUE ;
                      }
                      return FALSE ;

     case  WM_TIMER:
                      EditPrintf (hwndEdit, TEXT (".")) ;
                      return TRUE ;
```

```
 case  WM_SOCKET_NOTIFY:
                  wEvent = WSAGETSELECTEVENT (lParam) ;   // ie, LOWORD
                  wError = WSAGETSELECTERROR (lParam) ;   // ie, HIWORD

                  // Process two events specified in WSAAsyncSelect

                  switch (wEvent)
                  {
  // This event occurs as a result of the "connect" call

                  case  FD_CONNECT:
                                   EditPrintf (hwndEdit, TEXT ("\r\n")) ;

                                   if (wError)
                                   {
 EditPrintf (     hwndEdit, TEXT ("Connect error #%i."), wError) ;
 SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
return TRUE ;
    }
  EditPrintf (hwndEdit, TEXT ("Connected to %hs.\r\n"), szIPAddr) ;

   // Try to receive data. The call will generate an error
   // of WSAEWOULDBLOCK and an event of FD_READ

   recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
   EditPrintf (hwndEdit, TEXT ("Waiting to receive...")) ;
   return TRUE ;

   // This even occurs when the "recv" call can be made

            case  FD_READ:
                                   KillTimer (hwnd, ID_TIMER) ;
                                   EditPrintf (hwndEdit, TEXT ("\r\n")) ;

                                   if (wError)
                                   {
  EditPrintf (hwndEdit, TEXT ("FD_READ error #%i."), wError) ;
  SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
  return TRUE ;
  }
                                   // Get the time and swap the bytes

    iSize = recv (sock, (char *) &ulTime, 4, 0) ;
    ulTime = ntohl (ulTime) ;
    EditPrintf (hwndEdit,
    TEXT ("Received current time of %u seconds ")
    TEXT ("since Jan. 1 1900.\r\n"), ulTime) ;
```

```
        // Change the system time

                                        ChangeSystemTime (hwndEdit, ulTime) ;
                                    SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
                                        return TRUE ;
                        }
                        return FALSE ;
      }
      return FALSE ;
}

BOOL CALLBACK ServerDlg (   HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
      static char *    szServer ;
      static WORD           wServer = IDC_SERVER1 ;
      char                                  szLabel [64] ;

      switch (message)
      {
      case  WM_INITDIALOG:
                        szServer = (char *) lParam ;
                CheckRadioButton (hwnd, IDC_SERVER1, IDC_SERVER10, wServer) ;
                        return TRUE ;

      case  WM_COMMAND:
                        switch (LOWORD (wParam))
                        {
                        case  IDC_SERVER1:
                        case  IDC_SERVER2:
                        case  IDC_SERVER3:
                        case  IDC_SERVER4:
                        case  IDC_SERVER5:
                        case  IDC_SERVER6:
                        case  IDC_SERVER7:
                        case  IDC_SERVER8:
                        case  IDC_SERVER9:
                        case  IDC_SERVER10:
                                        wServer = LOWORD (wParam) ;
                                        return TRUE ;

                        case  IDOK:
                                        GetDlgItemTextA    (hwnd,    wServer,
szLabel, sizeof (szLabel)) ;
                                        strtok (szLabel, "(") ;
                                        strcpy (szServer, strtok (NULL, ")")) ;
                                        EndDialog (hwnd, TRUE) ;
```

```
                                        return TRUE ;

                        case  IDCANCEL:
                                        EndDialog (hwnd, FALSE) ;
                                        return TRUE ;
                        }
                        break ;
      }
      return FALSE ;
}

void ChangeSystemTime (HWND hwndEdit, ULONG ulTime)
{
      FILETIME                          ftNew ;
      LARGE_INTEGER          li ;
      SYSTEMTIME                  stOld, stNew ;

      GetLocalTime (&stOld) ;
      stNew.wYear                                          = 1900 ;
      stNew.wMonth                                   = 1 ;
      stNew.wDay                                     = 1 ;
      stNew.wHour                                    = 0 ;
      stNew.wMinute                                  = 0 ;
      stNew.wSecond                                  = 0 ;
      stNew.wMilliseconds                = 0 ;

      SystemTimeToFileTime (&stNew, &ftNew) ;
      li = * (LARGE_INTEGER *) &ftNew ;
      li.QuadPart += (LONGLONG) 10000000 * ulTime ;
      ftNew = * (FILETIME *) &li ;
      FileTimeToSystemTime (&ftNew, &stNew) ;

      if (SetSystemTime (&stNew))
      {
                  GetLocalTime (&stNew) ;
                  FormatUpdatedTime (hwndEdit, &stOld, &stNew) ;
      }
      else
                  EditPrintf (hwndEdit, TEXT ("Could NOT set new date and time.")) ;
}

void FormatUpdatedTime (     HWND hwndEdit, SYSTEMTIME * pstOld, SYSTEMTIME *
pstNew)
{
      TCHAR szDateOld [64], szTimeOld [64], szDateNew [64], szTimeNew [64] ;
      GetDateFormat     (LOCALE_USER_DEFAULT,     LOCALE_NOUSEROVERRIDE     |
DATE_SHORTDATE,
                                          pstOld, NULL, szDateOld, sizeof
```

```
(szDateOld)) ;
      GetTimeFormat (  LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE |
                     TIME_NOTIMEMARKER | TIME_FORCE24HOURFORMAT,
                  pstOld, NULL, szTimeOld, sizeof (szTimeOld)) ;

      GetDateFormat     (LOCALE_USER_DEFAULT,     LOCALE_NOUSEROVERRIDE     |
DATE_SHORTDATE,
                  pstNew, NULL, szDateNew, sizeof (szDateNew)) ;
      GetTimeFormat (  LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE |
                     TIME_NOTIMEMARKER | TIME_FORCE24HOURFORMAT,
                  pstNew, NULL, szTimeNew, sizeof (szTimeNew)) ;

      EditPrintf (hwndEdit, TEXT ("System date and time successfully changed ")
                                TEXT    ("from\r\n\t%s,    %s.%03i    to\r\n\t%s,
%s.%03i."),
                  szDateOld, szTimeOld, pstOld->wMilliseconds,
                  szDateNew, szTimeNew, pstNew->wMilliseconds) ;
}

void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...)
{
      TCHAR            szBuffer [1024] ;
      va_list     pArgList ;

      va_start (pArgList, szFormat) ;
      wvsprintf (szBuffer, szFormat, pArgList) ;
      va_end (pArgList) ;
      SendMessage (hwndEdit, EM_SETSEL, (WPARAM) -1, (LPARAM) -1) ;
      SendMessage (hwndEdit, EM_REPLACESEL, FALSE, (LPARAM) szBuffer) ;
      SendMessage (hwn dEdit, EM_SCROLLCARET, 0, 0) ;
}
```

NETTIME.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

////////////////////////////////////////////////////////////////////////////
/
// Dialog
SERVERS DIALOG DISCARDABLE  20, 20, 274, 202
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "NIST Time Service Servers"
FONT 8, "MS Sans Serif"
BEGIN
      DEFPUSHBUTTON          "OK",IDOK,73,181,50,14
      PUSHBUTTON                  "Cancel",IDCANCEL,150,181,50,14
      CONTROL
                  "time-a.timefreq.bldrdoc.gov (132.163.135.130) NIST, Boulder,
```

```
Colorado",
                        IDC_SERVER1,"Button",BS_AUTORADIOBUTTON,9,7,256,16
      CONTROL
            "time-b.timefreq.bldrdoc.gov  (132.163.135.131)  NIST,  Boulder,
Colorado",
                  IDC_SERVER2,"Button",BS_AUTORADIOBUTTON,9,24,256,16
      CONTROL
                  "time-c.timefreq.bldrdoc.gov   (132.163.135.132)   Boulder,
Colorado",
                        IDC_SERVER3,"Button",BS_AUTORADIOBUTTON,9,41,256,16
      CONTROL
                  "utcnist.colorado.edu (128.138.140.44) University of Colorado,
Boulder",
                        IDC_SERVER4,"Button",BS_AUTORADIOBUTTON,9,58,256,16
      CONTROL
                        "time.nist.gov (192.43.244.18) NCAR, Boulder, Colorado",
                  IDC_SERVER5,"Button",BS_AUTORADIOBUTTON,9,75,256,16
      CONTROL
                        "time-a.nist.gov  (129.6.16.35)  NIST,  Gaithersburg,
Maryland",
                        IDC_SERVER6,"Button",BS_AUTORADIOBUTTON,9,92,256,16
      CONTROL
                        "time-b.nist.gov  (129.6.16.36)  NIST,  Gaithersburg,
Maryland",
                        IDC_SERVER7,"Button",BS_AUTORADIOBUTTON,9,109,256,16
      CONTROL
                        "time-nw.nist.gov  (131.107.1.10)  Microsoft,  Redmond,
Washington",
                        IDC_SERVER8,"Button",BS_AUTORADIOBUTTON,9,126,256,16
      CONTROL
                        "utcnist.reston.mci.net  (204.70.131.13)  MCI,  Reston,
Virginia",
                        IDC_SERVER9,"Button",BS_AUTORADIOBUTTON,9,143,256,16

      CONTROL
                              "nist1.data.com  (209.0.72.7)  Datum,  San  Jose,
California",
                        IDC_SERVER10,"Button",BS_AUTORADIOBUTTON,9,160,256,16
END
NETTIME DIALOG DISCARDABLE  0, 0, 270, 150 STYLE WS_CHILD FONT 8, "MS Sans Serif"
BEGIN
      DEFPUSHBUTTON          "Set Correct Time",IDOK,95,129,80,14
      PUSHBUTTON                   "Close",IDC_CLOSE,183,129,80,14
      PUSHBUTTON                   "Select Server...",IDC_SERVER,7,129,80,14
      EDITTEXT                           IDC_TEXTOUT,7,7,253,110,ES_MULTILINE
| ES_AUTOVSCROLL |
                                                   ES_READONLY | WS_VSCROLL |
NOT WS_TABSTOP
```

```
END
 RESOURCE.H （摘录）
// Microsoft Developer Studio generated include file.
// Used by NetTime.rc

#define          IDC_TEXTOUT                                    101
#define          IDC_SERVER1                                    1001
#define          IDC_SERVER2                                    1002
#define          IDC_SERVER3                                    1003
#define          IDC_SERVER4                                    1004
#define          IDC_SERVER5                                    1005
#define          IDC_SERVER6                                    1006
#define          IDC_SERVER7                                    1007
#define          IDC_SERVER8                                    1008
#define          IDC_SERVER9                                    1009
#define          IDC_SERVER10                                   1010
#define          IDC_SERVER                                     1011
#define          IDC_CLOSE                                      1012
```

在结构上，NETTIME 程式建立了一个依据 NETTIME.RC 中的 NETTIME 所建立的非系统模态对话方块。程式重新定义了视窗的尺寸，以便非系统模态对话方块可以覆盖程式的整个视窗显示区域。对话方块包括一个唯读编辑区（程式用於写入文字资讯）、一个「Select Server」按钮、一个「Set Correct Time」按钮和一个「Close」按钮。「Close」按钮用於终止程式。

MainDlg 中的 szIPAddr 变数用於储存伺服器位址，内定是字串「132.163.135.130」。「Select Server」按钮启动依据 NETTIME.RC 中的 SERVERS 模板建立的对话方块。szIPAddr 变数作为最後一个参数传递给 DialogBoxParam。「Server」对话方块列出了 10 个伺服器（都是从 NIST 网站上逐字复制来的），这些伺服器提供了我们感兴趣的服务。当使用者单击一个伺服器时，ServerDlg 将分析按钮文字，以获得相应的 IP 位址。新位址储存在 szIPAddr 变数中。

当使用者按下「Set Correct Time」按钮时，按钮将产生一个 WM_COMMAND 讯息，其中 wParam 的低字组等於 IDOK。MainDlg 中的 IDOK 处理是大部分 Socket 初始行为发生的地方。

使用 Windows Sockets API 时，任何 Windows 程式必须呼叫的第一个函式是：

```
iError = WSAStartup (wVersion, &WSAData) ;
```

NETTIME 将第一个参数设定为 0x0200（表示 2.0 版本）。传回时，WSAData 结构包含了 Windows Sockets 实作的相关资讯，而且 NETTIME 将显示 szDescription 字串，并简要提供了一些版本资讯。

然後，NETTIME 如下呼叫 socket 函式：

```
sock = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
```

第一个参数是一个位址种类，表示此处是某种 Internet 位址。第二个参数表示资料以资料流的形式传回，而不是以资料封包的形式传回（我们需要的资料只有 4 个位元组长，而资料封包适用於较大的资料块）。最後一个参数是一个协定，我们指定使用的 Internet 协定是 TCP。它是 RFC-868 所定义的两个协定之一。socket 函式的传回值储存在 SOCKET 型态的变数中，以便後面的 Socket 函式的呼叫。

NETTIME 下面呼叫的 WSAAsynchSelect 是另一个 Windows 特有的 Socket 函式。此函式用於避免因 Internet 回应过慢而造成应用程式当住。在 WinSock 文件中，有些函式与「阻碍性（blocking）」有关。也就是说，它们不能保证立即把控制项权传回给程式。WSAAsyncSelect 函式强制阻碍性的函式转为非阻碍性的，即在函式执行完之前把控制项传回给程式。函式的结果以讯息的形式报告给应用程式。WSAAsyncSelect 函式让应用程式指定讯息和接收讯息的视窗的数值。通常，函式的语法如下：

```
WSAAsyncSelect (sock, hwnd, message, iConditions) ;
```

为此任务，NETTIME 使用程式定义的一个讯息，该讯息称为 WM_SOCKET_NOTIFY。它也用 WSAAsyncSelect 的最後一个参数来指定讯息发送的条件，特别在连结和接收资料时（FD_CONNECT | FD_READ）。

NETTIME 呼叫的下一个 WinSock 函式是 connect。此函式需要一个指向 Socket 位址结构的指标，对於不同的协定来说，此 Socket 位址结构是不同的。NETTIME 使用为 TCP/IP 设计的结构版本：

```
struct sockaddr_in
{
      short                                   sin_family;
      u_short                              sin_port;
      struct in_addr    sin_addr;
      char                                    sin_zero[8];
} ;
```

其中 in_addr 是用於指定 Internet 位址，它可以用 4 个位元组，或者 2 个无正负号短整数，或者 1 个无正负号长整数来表示。

NETTIME 将 sin_family 栏位设定为 AF_INET，用於表示位址种类。将 sin_port 设定为埠号，这里是时间协定的埠号，RFC-868 显示为 37。但不要像我最初时那样，将此栏位设为 37。当大多数数字通过 Internet 时，结构的这个埠号栏位必须是「big endian」的，即最高的位元组排第一个。Intel 微处理器是 little endian。幸运的是，htons（「host-to-network short」）函式使位元组翻转，因此 NETTIME 将 sockaddr_in 结构的 sin_port 栏位设定为：

```
htons (IPPORT_TIMESERVER)
```

WINSOCK2.H 中将常数定义为 37。NETTIME 用 inet_addr 函式将储存在

szIPAddr 字串中的伺服器位址转化为无正负号长整数，该整数用於设定结构的 sin_addr 栏位。

如果应用程式在 Windows 98 下呼叫 connect，而且目前 Windows 没有连结到 Internet，那么将显示「拨号连线」对话方块。这就是所谓的「自动拨号」。在 Windows NT 4.0 中没有实作「自动拨号」，因此如果在 NT 环境下执行，那么在执行 NETTIME 之前，就必须先连结上 Internet。

connect 函式通常已经会阻碍著後面程式的执行，这是因为连结成功以前需要花些时间。然而，由於 NETTIME 呼叫了 WSAAsyncSelect，所以 connect 不会等待连结，事实上，它会立即传回 SOCKET_ERROR 的值。这并不是出现了错误，这只是表示现在还没有连线成功而已。NETTIME 也不会检查这个传回值，只是呼叫 WSAGetLastError 而已。如果 WSAGetLastError 传回 WSAEWOULDBLOCK（即函式的执行通常要受阻，但这里并没有受阻），那就一切都还很正常。NETTIME 将「Set Correct Time」按钮改成「Cancel」，并设定了一个 1 秒的计时器。WM_TIMER 的处理方式只是在程式视窗中显示句点，以告诉使用者程式仍在执行，系统没有当掉。

连结最终完成时，MainDlg 由 WM_SOCKET_NOTIFY 讯息——NETTIME 在 WSAAsyncSelect 函式中指定的程式自订讯息所通知。lParam 的低字组等於 FD_CONNECT，高字组表示错误。这时的错误可能是程式不能连结到指定的伺服器。NETTIME 还列出了其他 9 个伺服器，供您选择，让您可以试试其他的伺服器。

如果一切顺利，那么 NETTIME 将呼叫 recv（「receive：接收」）函式来读取资料：

```
recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
```

这意味著，用 4 个位元组来储存 ulTime 变数。最後一个参数表示只是读此资料，并不将其从输入伫列中删除。像 connect 函式一样，recv 传回一个错误代码，以表示函式通常受阻，但这时没有受阻。理论上来说（当然这不大可能），函式至少能传回资料的一部分，然後透过再次呼叫以获得其余的 32 个位元组值。那就是呼叫 recv 函式时带有 MSG_PEEK 选项的原因。

与 connect 函式类似，recv 函式也产生 WM_SOCKET_NOTIFY 讯息，这时带有 FD_READ 的事件代码。NETTIME 通过再次呼叫 recv 来对此回应，这时最後的参数是 0，用於从伫列中删除资料。我将简要讨论一下程式处理接收到的 ulTime 的方法。注意，NETTIME 通过向自己发送 WM_COMMAND 讯息来结束处理，该讯息中 wParam 等於 IDCANCEL。对话方块程序通过呼叫 closesocket 和 WSACleanup 来回应。

再次呼叫 NETTIME 接收的 32 位元的 ulTime 值是从 1990 年 1 月 1 日开始的

0:00 UTC 秒数。但最高顺序的位元组是第一个位元组，因此该值必须通过 ntohl（「network-to-host long」）函式处理来调整位元组顺序，以便 Intel 微处理器能够处理。然後，NETTIME 呼叫 ChangeSystemTime 函式。

ChangeSystemTime 首先取得目前的本地时间——即使用者所在时区和日光节约时间的目前系统时间。将 SYSTEMTIME 结构设定为 1900 年 1 月 1 日午夜（0 时）。并将这个 SYSTEMTIME 结构传递给 SystemTimeToFileTime，将此结构转化为 FILETIME 结构。FILETIME 实际上只是由两个 32 位元的 DWORD 一起组成 64 位元的整数，用来表示从 1601 年 1 月 1 日至今间隔为 100 奈秒（nanosecond）的间隔数。

ChangeSystemTime 函式将 FILETIME 结构转化为 LARGE_INTEGER。它是一个 union，允许 64 位元的值可以被当成两个 32 位元的值使用，或者当成一个 __int64 资料型态的 64 位元整数使用（__int64 资料型态是 Microsoft 编译器对 ANSI C 标准的扩充）。因此，此值是 1601 年 1 月 1 日到 1900 年 1 月 1 日之间间隔为 100 奈秒的间隔数。这里，添加了 1900 年 1 月 1 日至今间隔为 100 奈秒的间隔数——ulTime 的 10,000,000 倍。

然後通过呼叫 FileTimeToSystemTime 将作为结果的 FILETIME 值转换回 SYSTEMTIME 结构。因为时间协定传回目前的 UTC 时间，所以 NETTIME 通过呼叫 SetSystemTime 来设定时间，SetSystemTime 也依据 UTC。基於显示的目的，程式呼叫 GetLocalTime 来获得更新时间。最初的本地时间和新的本地时间一起传递给 FormatUpdatedTime，这个函式用 GetTimeFormat 函式和 GetDateFormat 函式将时间转化为 ASCII 字串。

如果程式在 Windows NT 下执行，并且使用者没有取得设定时间的许可权，那么 SetSystemTime 函式可能失败。如果 SetSystemTime 失败，则 NETTIME 将发出一个新时间未设定成功的讯息来指出问题所在。

WININET 和 FTP

WinInet（「Windows Internet」）API 是一个高阶函式集，帮助程式写作者使用三个常见的 Internet 协定，这三个协定是：用於 World Wide Web 全球资讯网的超文字传输协定（HTTP：Hypertext Transfer Protocol）、档案传输协定（FTP：File Transfer Protocol）和另一个称为 Gopher 的档案传输协定。WinInet 函式的语法与常用的 Windows 档案函式的语法类似，这使得使用这些协定就像使用本地磁碟机上的档案一样容易。WinInet API 的文件位於/Platform SDK/Internet, Intranet, Extranet Services/Internet Tools and Technologies/WinInet API。

下面的范例程式将展示如何使用 WinInet API 的 FTP 部分。许多有网站的公司也都有「匿名 FTP」伺服器，这样使用者可以在不输入使用者名称和密码的情况下下载档案。例如，如果您在 Internet Explorer 的位址栏输入 ftp://ftp.microsoft.com ，那么您就可以浏览 FTP 伺服器上的目录并下载档案。如果进入 ftp://ftp.cpetzold.com/cpetzold.com/ProgWin/UpdDemo，那么您将在我的匿名 FTP 伺服器上发现与待会要提到的范例程式一块使用的档案列表。

虽然现今 FTP 服务对大多数的 Web 使用者来说并不是那么方便使用，但它仍然相当有用。例如，应用程式能利用 FTP 从匿名 FTP 伺服器上取得资料，这些取得资料的运作程序几乎完全在台面下处理，而不需要使用者操心。这就是我们将讨论的 UPDDEMO（「update demonstration：更新范例」）程式的构想。

FTP API 概况

使用 WinInet 的程式必须在所有呼叫 WinInet 函式的原始档案中包括表头档案 WININET.H。程式还必须连结 WININET.LIB。在 Microsoft Visual C++中，您可以在「Project Settings」对话方块的「Link」页面标签中指定。执行时，程式将和 WININET.DLL 动态连结程式库连结。

在下面的论述中，我不会详细讨论函式的语法，因为某些函式有很多选项，这让它变得相当复杂。要掌握 WinInet，您可以将 UPDDEMO 原始码当成食谱来看待。这时最重要的是了解有关的各个步骤以及 FTP 函式的范围。

要使用 Windows Internet API，首先要呼叫 InternetOpen。然後，使用 WinInet 支援的任何一种协定。InternetOpen 给您一个 Internet 作业代号，并储存到 HINTERNET 型态的变数中。用完 WinInet API 以後，应该通过呼叫 InternetCloseHandle 来关闭代号。

要使用 FTP，您接下来就要呼叫 InternetConnect。此函式需要使用由 InternetOpen 建立 Internet 作业代号，并且传回 FTP 作业的代号。您可将此代号作为名称开头为 Ftp 的所有函式的第一个参数。InternetConnect 函式的参数指出要使用的 FTP，还提供了伺服器名称，例如，ftp.cpetzold.com。此函式还需要使用者名称和密码。如果存取匿名 FTP 伺服器，这些参数可以设定为 NULL。如果应用程式呼叫 InternetConnect 时 PC 并没有连结到 Internet，Windows 98 将显示「拨号连线」对话方块。当使用 FTP 的应用程式结束时，呼叫 InternetCloseHandle 来关闭代号。

这时可以开始呼叫有 Ftp 字首的函式。您将发现这些函式与标准的 Windows 档案 I/O 函式很相似。为了避免与其他协定重复，一些以 Internet 为字首的函

式也可以处理 FTP。

下面四个函式用於处理目录：

```
fSuccess = FtpCreateDirectory           (hFtpSession,    szDirectory) ;
fSuccess = FtpRemoveDirectory           (hFtpSession,    szDirectory) ;
fSuccess = FtpSetCurrentDirectory (hFtpSession,    szDirectory) ;
fSuccess = FtpGetCurrentDirectory (hFtpSession,    szDirectory,
&dwCharacterCount) ;
```

注意，这些函式很像我们所熟悉的 Windows 提供用於处理本地档案系统的 CreateDirectory 、 RemoveDirectory 、 SetCurrentDirectory 和 GetCurrentDirectory 函式。

当然，存取匿名 FTP 的应用程式不能建立或删除目录。而且，程式也不能假定 FTP 目录具有和 Windows 档案系统相同的目录结构型态。特别是用相对路径名设定目录的程式，不能假定关於新的目录全名的一切。如果程式需要知道最後所在目录的整个名称，那么呼叫了 SetCurrentDirectory 之後必须再呼叫 GetCurrentDirectory。GetCurrentDirectory 的字串参数至少包含 MAX_PATH 字元，并且最後一个参数应指向包含该值的变数。

下面两个函式让您删除或者重新命名档案（但不是在匿名 FTP 伺服器上）：

```
fSuccess = FtpDeleteFile (hFtpSession, szFileName) ;
fSuccess = FtpRenameFile (hFtpSession, szOldName, szNewName) ;
```

经由先呼叫 FtpFindFirstFile，可以查找档案（或与含有万用字元的档名样式相符的多个档案）。此函式很像 FindFirstFile 函式，甚至都使用了相同的 WIN32_FIND_DATA 结构。该档案为列举出来的档案传回了一个代号。您可以将此代号传递给 InternetFindNextFile 函式以获得额外的档案名称资讯。最後通过呼叫 InternetCloseHandle 来关闭代号。

要打开档案，可以呼叫 FtpFileOpen。这个函式传回一个档案代号，此代号可以用於 InternetReadFile 、 InternetReadFileEx 、 InternetWrite 和 InternetSetFilePointer 呼叫。最後可以通过呼叫最常用的 InternetCloseHandle 函式来关闭代号。

最後，下面两个高级函式特别有用：FtpGetFile 呼叫将档案从 FTP 伺服器复制到本地记忆体，它合并了 FtpFileOpen、FileCreate、InternetReadFile、WriteFile、InternetCloseHandle 和 CloseHandle 呼叫。FtpGetFile 的另一个参数是一个旗标，如果本地已经存在同名档案，那么该旗标将导致函式呼叫失败。FtpPutFile 与此函式类似，用於将档案从本地记忆体复制到 FTP 伺服器。

更新展示程式

UPDDEMO，如程式 23-2 所示，展示了用 WinInet FTP 函式在第二个执行绪

执行期间从匿名 FTP 伺服器上下载档案的方法。

程式 23-2 UPDDEMO

```
UPDDEMO.C
/*------------------------------------------------------------------------
-
      UPDDEMO.C -- Demonstrates Anonymous FTP Access
                                                (c) Charles Petzold, 1998
------------------------------------------------------------------------
*/

#include         <windows.h>
#include         <wininet.h>
#include         <process.h>
#include         "resource.h"

            // User-defined messages used in WndProc

#define     WM_USER_CHECKFILES      (WM_USER + 1)
#define     WM_USER_GETFILES              (WM_USER + 2)

            // Information for FTP download

#define     FTPSERVER   TEXT  ("ftp.cpetzold.com")
#define     DIRECTORY   TEXT  ("cpetzold.com/ProgWin/UpdDemo")
#define     TEMPLATE    TEXT  ("UD??????.TXT")

    // Structures used for storing filenames and contents
typedef struct
{
      TCHAR * szFilename ;
      char  * szContents ;
}
FILEINFO ;
typedef struct
{
      int                     iNum ;
      FILEINFO          info[1] ;
}
FILELIST ;
    // Structure used for second thread
typedef struct
{
      BOOL bContinue ;
      HWND hwnd ;
}
PARAMS ;
            // Declarations of all functions in program
```

```
LRESULT     CALLBACK         WndProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL             CALLBACK         DlgProc (HWND, UINT, WPARAM, LPARAM) ;
VOID                                    FtpThread (PVOID) ;
VOID                                    ButtonSwitch (HWND, HWND, TCHAR *) ;
FILELIST *                         GetFileList (VOID) ;
int                                Compare (const FILEINFO *, const FILEINFO *) ;

            // A couple globals

HINSTANCE hInst ;
TCHAR                  szAppName[] = TEXT ("UpdDemo") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                                    PSTR    szCmdLine,    int
iCmdShow)
{
      HWND                                  hwnd ;
      MSG                                   msg ;
      WNDCLASS                              wndclass ;

      hInst = hInstance ;
      wndclass.style                        = 0 ;
      wndclass.lpfnWndProc                  = WndProc ;
      wndclass.cbClsExtra                   = 0 ;
      wndclass.cbWndExtra                   = 0 ;
      wndclass.hInstance                    = hInstance ;
      wndclass.hIcon                        = LoadIcon (NULL, IDI_APPLICATION) ;
      wndclass.hCursor                      = NULL ;
      wndclass.hbrBackground       = GetStockObject (WHITE_BRUSH) ;
      wndclass.lpszMenuName        = NULL ;
      wndclass.lpszClassName       = szAppName ;

      if (!RegisterClass (&wndclass))
      {
            MessageBox (     NULL, TEXT ("This program requires Windows NT!"),
                                                    szAppName, MB_ICONERROR) ;
                  return 0 ;
    }

      hwnd = CreateWindow (  szAppName, TEXT ("Update Demo with Anonymous FTP"),
                  WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL, NULL, hInstance, NULL) ;

      ShowWindow (hwnd, iCmdShow) ;
      UpdateWindow (hwnd) ;
```

```
                // After window is displayed, check if the latest file exists
     SendMessage (hwnd, WM_USER_CHECKFILES, 0, 0) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (   HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
     static FILELIST *plist ;
     static int                         cxClient, cyClient, cxChar, cyChar ;
     HDC                                           hdc ;
     int                                           i ;
     PAINTSTRUCT                        ps ;
     SCROLLINFO                         si ;
     SYSTEMTIME                         st ;
     TCHAR                              szFilename [MAX_PATH] ;

     switch (message)
    {
     case  WM_CREATE:
                    cxChar = LOWORD (GetDialogBaseUnits ()) ;
                    cyChar = HIWORD (GetDialogBaseUnits ()) ;
                    return 0 ;

     case  WM_SIZE:
                    cxClient         = LOWORD (lParam) ;
                    cyClient         = HIWORD (lParam) ;

                    si.cbSize   = sizeof (SCROLLINFO) ;
                    si.fMask         = SIF_RANGE | SIF_PAGE ;
                    si.nMin          = 0 ;
                    si.nMax          = plist ? plist->iNum - 1 : 0 ;
                    si.nPage         = cyClient / cyChar ;

                    SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                    return 0 ;

     case  WM_VSCROLL:
                    si.cbSize   = sizeof (SCROLLINFO) ;
                    si.fMask         = SIF_POS | SIF_RANGE | SIF_PAGE ;
                    GetScrollInfo (hwnd, SB_VERT, &si) ;

                    switch (LOWORD (wParam))
```

```
                    {
                    case SB_LINEDOWN: si.nPos += 1 ; break ;
                    case SB_LINEUP:   si.nPos -= 1 ; break ;
                    case SB_PAGEDOWN: si.nPos += si.nPage ; break ;
                    case SB_PAGEUP:   si.nPos -= si.nPage ; break ;
                    case  SB_THUMBPOSITION:  si.nPos  =  HIWORD  (wParam)  ;
     break ;
                    default:               return 0 ;
                    }
                    si.fMask = SIF_POS ;
                    SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

     case  WM_USER_CHECKFILES:
                    // Get the system date & form filename from year and month

                    GetSystemTime (&st) ;
          wsprintf (szFilename, TEXT ("UD%04i%02i.TXT"), st.wYear, st.wMonth) ;

                    // Check if the file exists; if so, read all the files

                    if (GetFileAttributes (szFilename) != (DWORD) -1)
                    {
                              SendMessage (hwnd, WM_USER_GETFILES, 0, 0) ;
                              return 0 ;
                    }
                    // Otherwise, get files from Internet.
                    // But first check so we don't try to copy files to a CD-ROM!

                    if (GetDriveType (NULL) == DRIVE_CDROM)
                    {
   MessageBox (hwnd, TEXT ("Cannot run this program from CD-ROM!"),
             szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                              return 0 ;
                    }
                              // Ask user if an Internet connection is desired

                    if (IDYES == MessageBox (hwnd, TEXT ("Update information
from Internet?"),
                    szAppName, MB_YESNO | MB_ICONQUESTION))

                                        // Invoke dialog box

                DialogBox (hInst, szAppName, hwnd, DlgProc) ;

                                        // Update display
```

```
                    SendMessage (hwnd, WM_USER_GETFILES, 0, 0) ;
                    return 0 ;

     case  WM_USER_GETFILES:
                    SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
                    ShowCursor (TRUE) ;

                                        // Read in all the disk files

                    plist = GetFileList () ;

                    ShowCursor (FALSE) ;
                    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                    // Simulate a WM_SIZE message to alter scroll bar & repaint

                SendMessage (hwnd, WM_SIZE, 0, MAKELONG (cxClient, cyClient)) ;
                    InvalidateRect (hwnd, NULL, TRUE) ;
                    return 0 ;

     case  WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps) ;
                    SetTextAlign (hdc, TA_UPDATECP) ;

                    si.cbSize = sizeof (SCROLLINFO) ;
                    si.fMask  = SIF_POS ;
                    GetScrollInfo (hwnd, SB_VERT, &si) ;

                    if (plist)
                    {
                                for (i = 0 ; i < plist->iNum ; i++)
                                {
               MoveToEx    (hdc, cxChar, (i - si.nPos) * cyChar, NULL) ;
               TextOut     (hdc, 0, 0, plist->info[i].szFilename,
               lstrlen (plist->info[i].szFilename)) ;
               TextOut  (hdc, 0, 0, TEXT (": "), 2) ;
               TextOutA (hdc, 0, 0, plist->info[i].szContents,
               strlen (plist->info[i].szContents)) ;
                                }
                    }
                    EndPaint (hwnd, &ps) ;
                    return 0 ;

     case  WM_DESTROY:
                    PostQuitMessage (0) ;
                    return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
```

```
}

BOOL CALLBACK DlgProc (      HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
      static PARAMS params ;
      switch (message)
      {
      case  WM_INITDIALOG:
                        params.bContinue = TRUE ;
                        params.hwnd = hwnd ;

                        _beginthread (FtpThread, 0, ¶ms) ;
                        return TRUE ;

      case  WM_COMMAND:
                        switch (LOWORD (wParam))
                        {
                        case  IDCANCEL:   // button for user to abort download
                                    params.bContinue = FALSE ;
                                    return TRUE ;

                        case  IDOK:     // button to make dialog box go away
                                    EndDialog (hwnd, 0) ;
                                    return TRUE ;
                        }
      }
      return FALSE ;
}

/*-------------------------------------------------------------------------
-
      FtpThread: Reads files from FTP server and copies them to local disk
-------------------------------------------------------------------------
-*/

void FtpThread (PVOID parg)
{
      BOOL                                    bSuccess ;
      HINTERNET                         hIntSession, hFtpSession, hFind ;
      HWND                                    hwndStatus, hwndButton ;
      PARAMS                            *     pparams ;
      TCHAR                                   szBuffer [64] ;
      WIN32_FIND_DATA         finddata ;

      pparams           = parg ;
      hwndStatus  = GetDlgItem (pparams->hwnd, IDC_STATUS) ;
      hwndButton  = GetDlgItem (pparams->hwnd, IDCANCEL) ;
```

```
                    // Open an internet session

     hIntSession = InternetOpen (szAppName, INTERNET_OPEN_TYPE_PRECONFIG,
                          NULL, NULL, INTERNET_FLAG_ASYNC) ;
     if (hIntSession == NULL)
     {
      wsprintf (szBuffer, TEXT ("InternetOpen error %i"), GetLastError ()) ;
               ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
               _endthread () ;
     }

     SetWindowText (hwndStatus, TEXT ("Internet session opened...")) ;
                    // Check if user has pressed Cancel
     if (!pparams->bContinue)
     {
               InternetCloseHandle (hIntSession) ;
               ButtonSwitch (hwndStatus, hwndButton, NULL) ;
               _endthread () ;
     }

               // Open an FTP session.
hFtpSession      =      InternetConnect      (hIntSession,      FTPSERVER,
INTERNET_DEFAULT_FTP_PORT,
                       NULL, NULL, INTERNET_SERVICE_FTP, 0, 0) ;
     if (hFtpSession == NULL)
     {
               InternetCloseHandle (hIntSession) ;
               wsprintf (szBuffer, TEXT ("InternetConnect error %i"),
                                                    GetLastError ()) ;
          ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
          _endthread () ;
    }

     SetWindowText (hwndStatus, TEXT ("FTP Session opened...")) ;
                    // Check if user has pressed Cancel
     if (!pparams->bContinue)
     {
               InternetCloseHandle (hFtpSession) ;
               InternetCloseHandle (hIntSession) ;
               ButtonSwitch (hwndStatus, hwndButton, NULL) ;
               _endthread () ;
     }

                    // Set the directory
     bSuccess = FtpSetCurrentDirectory (hFtpSession, DIRECTORY) ;
     if (!bSuccess)
     {
```

```
                InternetCloseHandle (hFtpSession) ;
                InternetCloseHandle (hIntSession) ;
                wsprintf ( szBuffer, TEXT ("Cannot set directory to %s"),
                                                        DIRECTORY) ;
                ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
                _endthread () ;
     }

     SetWindowText (hwndStatus, TEXT ("Directory found...")) ;
                // Check if user has pressed Cancel
     if (!pparams->bContinue)
     {
                InternetCloseHandle (hFtpSession) ;
                InternetCloseHandle (hIntSession) ;
                ButtonSwitch (hwndStatus, hwndButton, NULL) ;
                _endthread () ;
     }

                     // Get the first file fitting the template
     hFind = FtpFindFirstFile (hFtpSession, TEMPLATE, &finddata, 0, 0) ;
     if (hFind == NULL)
     {
                InternetCloseHandle (hFtpSession) ;
                InternetCloseHandle (hIntSession) ;
          ButtonSwitch (hwndStatus, hwndButton, TEXT ("Cannot find files")) ;
                _endthread () ;
     }

     do
     {
                          // Check if user has pressed Cancel
          if (!pparams->bContinue)
                {
                          InternetCloseHandle (hFind) ;
                          InternetCloseHandle (hFtpSession) ;
                          InternetCloseHandle (hIntSession) ;
                          ButtonSwitch (hwndStatus, hwndButton, NULL) ;
                          _endthread () ;
                }
                     // Copy file from internet to local hard disk, but fail
                     // if the file already exists locally

                wsprintf   (szBuffer,   TEXT   ("Reading   file   %s..."),
finddata.cFileName) ;
                SetWindowText (hwndStatus, szBuffer) ;

                FtpGetFile (     hFtpSession,
            finddata.cFileName, finddata.cFileName, TRUE,
```

```
                FILE_ATTRIBUTE_NORMAL, FTP_TRANSFER_TYPE_BINARY, 0) ;
      }
      while (InternetFindNextFile (hFind, &finddata)) ;
      InternetCloseHandle (hFind) ;
      InternetCloseHandle (hFtpSession) ;
      InternetCloseHandle (hIntSession) ;

      ButtonSwitch(hwndStatus, hwndButton, TEXT("Internet Download Complete"));
}

/*-------------------------------------------------------------------------
--
      ButtonSwitch:  Displays final status message and changes Cancel to OK

-------------------------------------------------------------------------*/
VOID ButtonSwitch (HWND hwndStatus, HWND hwndButton, TCHAR * szText)
{
      if (szText)
                        SetWindowText (hwndStatus, szText) ;
      else
                        SetWindowText  (hwndStatus,  TEXT  ("Internet  Session
Cancelled")) ;
      SetWindowText (hwndButton, TEXT ("OK")) ;
      SetWindowLong (hwndButton, GWL_ID, IDOK) ;
}

/*-------------------------------------------------------------------------
-
      GetFileList: Reads files from disk and saves their names and contents
-------------------------------------------------------------------------
-*/

FILELIST * GetFileList (void)
{
      DWORD                                          dwRead ;
      FILELIST                              *     plist ;
      HANDLE                                       hFile, hFind ;
      int                                               iSize, iNum  ;
      WIN32_FIND_DATA        finddata ;

      hFind = FindFirstFile (TEMPLATE, &finddata) ;
      if (hFind == INVALID_HANDLE_VALUE)
                        return NULL ;
      plist       = NULL ;
      iNum        = 0 ;

      do
      {
```

```
                                    // Open the file and get the size
                hFile   =   CreateFile   (finddata.cFileName,   GENERIC_READ,
FILE_SHARE_READ,
                   NULL, OPEN_EXISTING, 0, NULL) ;

                if (hFile == INVALID_HANDLE_VALUE)
                                    continue ;

                iSize = GetFileSize (hFile, NULL) ;

                if (iSize == (DWORD) -1)
           {
                                    CloseHandle (hFile) ;
                                    continue ;
                }
                               // Realloc the FILELIST structure for a new entry

           plist = realloc (plist, sizeof (FILELIST) + iNum * sizeof (FILEINFO));

                                    // Allocate space and save the filename

                plist->info[iNum].szFilename     =     malloc     (lstrlen
(finddata.cFileName) +sizeof (TCHAR)) ;
                lstrcpy (plist->info[iNum].szFilename, finddata.cFileName) ;

                                    // Allocate space and save the contents

                plist->info[iNum].szContents = malloc (iSize + 1) ;
        ReadFile (hFile, plist->info[iNum].szContents, iSize, &dwRead, NULL);
                plist->info[iNum].szContents[iSize] = 0 ;

                CloseHandle (hFile) ;
                iNum ++ ;
      }
      while (FindNextFile (hFind, &finddata)) ;
      FindClose (hFind) ;
                               // Sort the files by filename
      qsort (plist->info, iNum, sizeof (FILEINFO), Compare) ;
      plist->iNum = iNum ;
      return plist ;
}

/*-------------------------------------------------------------------------
-
      Compare function for qsort
-------------------------------------------------------------------------
*/
```

```
int Compare (const FILEINFO * pinfo1, const FILEINFO * pinfo2)
{
      return lstrcmp (pinfo2->szFilename, pinfo1->szFilename) ;
}
```

UPDDEMO.RC （摘录）

```
//Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
/
// Dialog
UPDDEMO DIALOG DISCARDABLE  20, 20, 186, 95
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Internet Download"
FONT 8, "MS Sans Serif"
BEGIN
      PUSHBUTTON             Cancel",IDCANCEL,69,74,50,14
      CTEXT                              "",IDC_STATUS,7,29,172,21
END
```

RESOURCE.H （摘录）

```
// Microsoft Developer Studio generated include file.
// Used by UpdDemo.rc

#define IDC_STATUS    40001
```

UPDDEMO 使用的档案名称是 UDyyyymm.TXT，其中 yyyy 是 4 位阿拉伯数字的年数（当然适用於 2000），mm 是 2 位阿拉伯数字的月数。这里假定程式可以享有每个月都有更新档案的好处。这些档案可能是整个月刊，而由於阅读效率上的考虑，让程式将其下载到本地储存媒体上。

因此，WinMain 在呼叫 ShowWindow 和 UpdateWindow 来显示 UPDDEMO 主视窗以後，向 WndProc 发送程式定义的 WM_USER_CHECKFILES 讯息。WndProc 通过获得目前的年、月并检查该年月 UDyyyymm.TXT 档案所在的内定目录来处理此讯息。这种档案的存在意义在於 UPDDEMO 会被完全更新（当然，事实并非如此。一些过时的档案将漏掉。如果要做得更完整，程式得进行更广泛的检测）。在这种情况下，UPDDEMO 向自己发送一个 WM_USER_GETFILES 讯息，它通过呼叫 GetFileList 函式来处理。这是 UPDDEMO.C 中稍长的一个函式，但它并不是特别有用，它所做的全部工作就是将所有的 UDyyyymm.TXT 档案读到动态配置的 FILELIST 型态结构中，该结构是在程式顶部定义的，然後让程式在其显示区域显示这些档案的内容。

如果 UPDDEMO 没有最新的档案，那么它必须透过 Internet 进行更新。程式首先询问使用者这样做是否「OK」。如果是，程式将显示一个简单的对话方块，

其中只有一个「Cancel」按钮和一个 ID 为 IDC_STATUS 的静态文字区。下载时，此静态文字区向使用者提供状态报告，并且允许使用者取消过於缓慢的更新作业。此对话程序的名称是 DlgProc。

DlgProc 很短，它建立了一个包括自身视窗代号的 PARAMS 型态的结构以及一个名称为 bContinue 的 BOOL 变数，然後呼叫_ beginthread 来执行第二个执行绪。

FtpThread 函式透过使用下面的呼叫来完成实际的传输：InternetOpen、InternetConnect 、 FtpSetCurrentDirectory 、 FtpFindFirstFile 、InternetFindNextFile、FtpGetFile 和 InternetCloseHandle（三次）。如同大多数程式码，该执行绪函式如果略过错误检查、让使用者了解下一步的操作情况以及允许使用者随意取消整个显示的那些步骤，那么它将变得简洁许多。FtpThread 函式透过用 hwndStatus 代号呼叫 SetWindowText 来让使用者知道进展情况，这里指的是对话方块中间的静态文字区。

执行绪可以依照下面的三种方式之一来终止：

第一种，FtpThread 可能遇到从 WinInet 函式传回的错误。如果是这样，它将清除并编排错误字串的格式，然後将此字串（连同对话方块文字区代号和「Cancel」按钮的代号一起）传递给 ButtonSwitch。ButtonSwitch 是一个小函式，它显示了文字字串，并将「Cancel」按钮转换成「OK」按钮——不只是按钮上的文字字串的转换，还包括控制项 ID 的转换。这样就允许使用者按下「OK」按钮来结束对话方块。

第二种方式，FtpThread 能在没有任何错误的情况下完成任务，其处理方法和遇到错误时的方法一样，只不过对话方块中显示的字串为「Internet Download Complete」。

第三种方式，使用者可以在程序中选择取消下载。这时，DlgProc 将 PARAMS 结构的 bContinue 栏位设定为 FALSE。FtpThread 频繁地检查该值，如果 bContinue 等於 FALSE，那么函式将做好应该进行的收拾工作，并以 NULL 文字参数呼叫 ButtonSwitch，此参数表示显示了字串「Internet Session Cancelled」。同样，使用者必须按下「OK」按钮来关闭对话方块。

虽然 UPDDEMO 取得的每个档案只能显示一行，但我（本书的作者）可以用这个程式来告诉您（本书的读者）本书的更新内容以及其他资讯，您也可以在网站上发现更详细的资讯。因此，UPDDEMO 成为我向您传送资讯的方法，并且可以让本书的内容延续到最後一页之後。