Please note that currently, this program will only work over a home network between 2 computers, not between 2 buildings. I'm working on it.

If you have problems with this code, please leave a comment on the discussion page. - spork222 Apr 13, 2006

IMPORTANT: This is a simple demo which uses only winsock calls (Windows built-in) and WMLiberty.dll for event trapping. WMLiberty can be found here.

HOW TO USE THIS PROGRAM.

Same computer:
1. Bring up 2 copies of LB.
2. Copy and paste the program into both of them.
3. In the first one, click run. Then, click the "Server" radiobutton, enter a display name, and click "Start."
4. The last message should say something like "Clients connect to [192.168.1.1]"
5. In the second copy of LB, run the program.
6. Copy and paste the address in block quotes from the first running program (Server) to the "IP Address" text box in the 2nd program.
7. Enter a display name, and click "Start."
8. Chat away!

2 computers:
1. On the first computer, bring up LB, copy and paste this program into the editor, and click Run.
2. Click the Server radiobutton, enter a display name, and click "Start".
3. Write down the address in block quotes (See step 4 above)
4. On the second computer, bring up LB, copy and paste this program into the editor, and click Run.
5. Type in the address you wrote down from above into the "IP Address" box.
6. Enter a display name.
7. Click "Start."
8. Start chatting!

Thanks to Brent Thorn who wrote the original program.

```
'*** Simple Server/Client Demo
'*** Phil, April 2006
'*** Original program by Brent Thorn, Feb. 2004
'This is a simple "chat" type application.
'You can set up a server, then connect, and
'both sides can talk back and forth.

nomainwin
WindowWidth = 185
WindowHeight = 230
```

```
UpperLeftX=int((DisplayWidth-WindowWidth)/2)
UpperLeftY=int((DisplayHeight-WindowHeight)/2)

groupbox #main.groupbox2, "Mode",   5,   7, 165,  45
TextboxColor$ = "white"
textbox #main.textbox1,   5,  77, 155,  25
radiobutton #main.radiobutton3, "Client", [radiobutton3Set], [radiobut
tonReset], 100,  22,  58,  25
radiobutton #main.radiobutton4, "Server", [radiobutton4Set], [radiobut
tonReset],  15,  22,  65,  25
statictext #main.statictext5, "IP Address",  10,  57, 144,  20
button #main.button7,"Start",[button7Click], UL,  55, 167,  50,  25
statictext #main.scname1, "Display name:", 5, 107, 100, 20
textbox #main.scrname2, 5, 127, 155, 25

open "Chat" for window_nf as #main
print #main, "font ms_sans_serif 10"
print #main, "trapclose [quit.main]"
print #main.radiobutton3, "set"
wait


[radiobutton3Set]
    server=0
    print #main.textbox1, "!enable"
    print #main.textbox1, "!show"
    print #main.statictext5, "!enable"
    wait

[radiobutton4Set]
    server=1
    print #main.textbox1, ""
    print #main.textbox1, "!hide"
    print #main.statictext5, "!disable"
    wait

[radiobuttonReset]
    wait

[quit.main]
    close #main
    end

[button7Click]
print #main.textbox1, "!contents? var$"
if server=0 and validip$(var$)="" then notice "Invalid IP Address":wai
```

```
t
print #main.scrname2, "!contents? name$"
if name$="" then notice "Please enter a display name":wait
close #main
addr$=validip$(var$)
for x=1 to 4
part$=word$(addr$,x)
part$=dechex$(val(part$))
if len(part$)=1 then part$="0"+part$
connaddr$=connaddr$+part$
next x
print connaddr$
connaddr=hexdec(connaddr$)

global accepts

    PORT = 4000

    NoMainWin

    Open "wsock32" For DLL As #wsock32
    Open "WMLiberty" For DLL As #wmlib

    ' Create a window.
    WindowWidth = 400
    WindowHeight = 400
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)

    TexteditorColor$ = "white"
    texteditor #s.te,  10,   2, 370, 275
    TextboxColor$ = "white"
    textbox #s.t,  10, 287, 370,  25
    button #s.s,"Send",[send], UL,  10, 317,  165,  25
    button #s.c,"Clear",[clear], UL, 215, 317,  165,  25

    open "Chat" for window_nf as #s
    print #s, "font ms_sans_serif 10"
    print #s.te, "!font system 10"
    print #s.te, "!autoresize"
    print #s.t, "!setfocus"
    print #s, "trapclose [s_Close]"


    ' Now create a socket, bind it to a local port, set some
    ' network events to trap, and start listening for clients.
```

```
    Call WinsockInit

    Err = 1 ' Assume failure
    If WSAStartup(MAKEWORD(2, 2)) = 0 Then
        #s.te "> Winsock initialized."

        sockaddr.sinfamily.struct = 2 'AF_INET
        sockaddr.sinzero.struct = String$(8, 0)
        sockaddr.sinport.struct = htons(PORT)
        If sockaddr.sinport.struct <> -1 Then
            sockaddr.sinaddr.struct = htonl(connaddr)
            If sockaddr.sinaddr.struct <> -1 Then
                sock = socket(2, 1, 0) 'AF_INET=2:SOCK_STREAM=1
                If sock <> -1 Then
                    #s.te "> Socket created."

                    if server=1 and bind(sock) = 0 Then #s.te "> Port
bind successful." else if server=1 then [breaknet]

                        Callback lpfnCB, SockProc( Long, Long, Long, L
ong ), Long
                        rc = SetWMHandler(HWnd(#s), _WM_USER, lpfnCB,
1)
                        'FD_READ=1:FD_WRITE=2:FD_OOB=4:FD_ACCEPT=8:FD_
CONNECT=16:FD_CLOSE=32
                        flags=1 or 2 or 8 or 32
                        if server=0 then flags=flags or 16
                        If WSAAsyncSelect(sock, HWnd(#s), _WM_USER, fl
ags) <> -1 Then
                            #s.te "> Events selected."

                            if server=1 and listen(sock, 1) = 0 Then #
s.te "> Listening for incoming connections.": Err = 0 ' Success!
                            If server=0 then
                            if connect(sock)=-1 and WSAGetLastError()=
10035 then
                                #s.te "> Connect requested."
                                Err=0
                            end if
                            end if
                        End If
                    [breaknet]
                End If
            End If
        End If
```

```
    End If

    If Err Then
        #s.te "> ERROR: "; GetWSAErrorString$(WSAGetLastError())
        If sock <> -1 Then
            rc = closesocket(sock)
        End If
    Else
        if server=1 then
            myip = GetLocalIP()
            #s.te "> Clients connect to ["; InetNtoA$(myip); "]"
        end if
    End If



[s_Wait]
    Scan
    CallDLL #kernel32, "Sleep", _
        100 As Long, _
        rc As Void
    GoTo [s_Wait]
[s_Close]
    Call WSACleanup

    Close #s

    Close #wmlib
    Close #wsock32

    End

[send]
    print #s.t, "!contents? var$"
    if server=1 then
        if Send(accepts,name$+"> "+var$+chr$(13),0)=-1 then #s.te "> E
RROR: "; GetWSAErrorString$(WSAGetLastError())
    else
        if Send(sock,name$+"> "+var$+chr$(13),0)=-1 then #s.te "> ERRO
R: "; GetWSAErrorString$(WSAGetLastError())
    end if
    #s.te name$+"> "+var$
    print #s.t, ""
    goto [s_Wait]

[clear]
```

```
    print #s.t, ""
    print #s.t, "!setfocus"
    goto [s_Wait]

'*** Application Procedures ***

Function SockProc( hWnd, uMsg, sock, lParam )
' Callback function to handle a Windows message
' forwarded by WMLiberty. Called when a relevant
' network event occurs.

    Select Case LOWORD(lParam)
        Case 1 'FD_READ
            buf$ = Recv$(sock, 8192, 0)
            While Len(buf$)
                #s.te woBang$(buf$);

                buf$ = Recv$(sock, 8192, 0)
            Wend
        Case 2 'FD_WRITE
            'TODO
        Case 8 'FD_ACCEPT
            accepts = accept(sock)
            #s.te ">Socket: ";accepts

            #s.te "> Accepted connection from "; _
                  InetNtoA$(sockaddr.sinaddr.struct); ":"; _
                  htons(sockaddr.sinport.struct); "."
        case 16 'FD_CONNECT
            if HIWORD(lParam)=0 then
                #s.te "> Connect complete."
                x=Send(sock,"Hi!"+chr$(13),0)
            else
                #s.te "> Connect failed."
            end if
        Case 32 'FD_CLOSE
            ' Flush the buffers.
            buf$ = Recv$(sock, 8192, 0)
            While Len(buf$)
                #s.te woBang$(buf$);

                buf$ = Recv$(sock, 8192, 0)
            Wend
            #s.te "> Connection Closed."
    End Select
SockProc=1
```

```
End Function

Sub WinsockInit
' Initializes structs used in Winsock calls.
    Struct hostent, _
        hname As Long, _
        haliases As Long, _
        haddrtype As Word, _
        hlength As Word, _
        haddrlist As Long

    Struct sockaddr, _
        sinfamily As Short, _
        sinport As UShort, _
        sinaddr As ULong, _
        sinzero As Char[8]

    Struct WSAData, _
        wVersion As Word, _
        wHighVersion As Word, _
        szDescription As Char[257], _
        szSystemStatus As Char[129], _
        iMaxSockets As Word, _
        iMaxUdpDg As Word, _
        lpVendorInfo As Long
End Sub

Function woBang$( raw$ )
' Kludge to print a string that could start with an
' exclamation point, or bang (!). Am I missing something?
    woBang$ = raw$
    bangs = 0
    While Mid$(raw$, bangs+1, 1) = "!"
        bangs = bangs + 1
    Wend
    If bangs Then
        bang$ = Left$(raw$, bangs)
        woBang$ = Mid$(raw$, bangs+1)

        #s.te "!Lines ln"
        #s.te "!Line "; ln; " ln$"
        #s.te "!Select "; Len(ln$)+1; " "; ln
        #s.te "!Insert bang$"
        #s.te "!Select 1 1"
    End If
End Function
```

```
'*** General Procedures ***

Function LOWORD( dw )
    LOWORD = (dw And 65535)
End Function

Function HIWORD( dw )
    HIWORD = int((dw / 65536))
End Function

Function MAKEWORD( b1, b2 )
    MAKEWORD = b1 Or (256 * b2)
End Function

Function String$( num, ch )
    If num > 0 Then
        String$ = Chr$(ch)
        While Len(String$) < num
            String$ = String$ + String$
        Wend
        String$ = Left$(String$, num)
    End If
End Function

'*** Winsock Wrappers ***

Function GetHostByAddr$( addr )
    Struct p, addr As ULong
    p.addr.struct = addr
    CallDLL #wsock32, "gethostbyaddr", _
        p As Struct, _
        4 As Long, _
        2 As Long, _ 'AF_INET=2
        phe As Long
    If phe Then
        helen = Len(hostent.struct)
        CallDLL #kernel32, "RtlMoveMemory", _
            hostent As Struct, _
            phe As ULong, _
            helen As Long, _
            rc As Void
        GetHostByAddr$ = WinString(hostent.hname.struct)
    End If
End Function
```

```
Function GetHostByName$( sName$ )
    CallDLL #wsock32, "gethostbyname", _
        sName$ As Ptr, _
        phe As ULong
    If phe Then
        helen = Len(hostent.struct)
        CallDLL #kernel32, "RtlMoveMemory", _
            hostent As Struct, _
            phe As ULong, _
            helen As Long, _
            rc As Void
        GetHostByName$ = WinString(hostent.hname.struct)
    End If
End Function

Function GetHostName$()
    buf$ = Space$(256)+Chr$(0)
    CallDLL #wsock32, "gethostname", _
        buf$ As Ptr, _
        256 As Long, _
        rc As Long
    GetHostName$ = Trim$(buf$)
End Function

Function GetLocalIP()
    sName$ = GetHostName$()
    CallDLL #wsock32, "gethostbyname", _
        sName$ As Ptr, _
        phe As ULong
    If phe Then
        helen = Len(hostent.struct)
        CallDLL #kernel32, "RtlMoveMemory", _
            hostent As Struct, _
            phe As ULong, _
            helen As Long, _
            rc As Void
        plong = hostent.haddrlist.struct
        Struct p, addrlist As ULong
        CallDLL #kernel32, "RtlMoveMemory", _
            p As Struct, _
            plong As ULong, _
            4 As Long, _
            rc As Void
        plong = p.addrlist.struct
        Struct p, addr As ULong
        hlength = hostent.hlength.struct
```

```
        CallDLL #kernel32, "RtlMoveMemory", _
            p As Struct, _
            plong As ULong, _
            hlength As Long, _
            rc As Void
        GetLocalIP = p.addr.struct
    End If
End Function

Function GetWSAErrorString$( errnum )
    Select Case errnum
        Case 10004: e$ = "Interrupted system call."
        Case 10009: e$ = "Bad file number."
        Case 10013: e$ = "Permission Denied."
        Case 10014: e$ = "Bad Address."
        Case 10022: e$ = "Invalid Argument."
        Case 10024: e$ = "Too many open files."
        Case 10035: e$ = "Operation would block."
        Case 10036: e$ = "Operation now in progress."
        Case 10037: e$ = "Operation already in progress."
        Case 10038: e$ = "Socket operation on nonsocket."
        Case 10039: e$ = "Destination address required."
        Case 10040: e$ = "Message too long."
        Case 10041: e$ = "Protocol wrong type for socket."
        Case 10042: e$ = "Protocol not available."
        Case 10043: e$ = "Protocol not supported."
        Case 10044: e$ = "Socket type not supported."
        Case 10045: e$ = "Operation not supported on socket."
        Case 10046: e$ = "Protocol family not supported."
        Case 10047: e$ = "Address family not supported by protocol fam
ily."
        Case 10048: e$ = "Address already in use."
        Case 10049: e$ = "Can't assign requested address."
        Case 10050: e$ = "Network is down."
        Case 10051: e$ = "Network is unreachable."
        Case 10052: e$ = "Network dropped connection."
        Case 10053: e$ = "Software caused connection abort."
        Case 10054: e$ = "Connection reset by peer."
        Case 10055: e$ = "No buffer space available."
        Case 10056: e$ = "Socket is already connected."
        Case 10057: e$ = "Socket is not connected."
        Case 10058: e$ = "Can't send after socket shutdown."
        Case 10059: e$ = "Too many references: can't splice."
        Case 10060: e$ = "Connection timed out."
        Case 10061: e$ = "Connection refused."
        Case 10062: e$ = "Too many levels of symbolic links."
```

```
        Case 10063: e$ = "File name too long."
        Case 10064: e$ = "Host is down."
        Case 10065: e$ = "No route to host."
        Case 10066: e$ = "Directory not empty."
        Case 10067: e$ = "Too many processes."
        Case 10068: e$ = "Too many users."
        Case 10069: e$ = "Disk quota exceeded."
        Case 10070: e$ = "Stale NFS file handle."
        Case 10071: e$ = "Too many levels of remote in path."
        Case 10091: e$ = "Network subsystem is unusable."
        Case 10092: e$ = "Winsock DLL cannot support this application.
"
        Case 10093: e$ = "Winsock not initialized."
        Case 10101: e$ = "Disconnect."
        Case 11001: e$ = "Host not found."
        Case 11002: e$ = "Nonauthoritative host not found."
        Case 11003: e$ = "Nonrecoverable error."
        Case 11004: e$ = "Valid name, no data record of requested type
."
        Case Else:  e$ = "Unknown error "; errnum; "."
    End Select
    GetWSAErrorString$ = e$
End Function

Function InetNtoA$( inaddr )
    CallDLL #wsock32, "inet_ntoa", _
        inaddr As ULong, _
        pstr As ULong
    InetNtoA$ = WinString(pstr)
End Function

Function Recv$( s, buflen, flags )
    Recv$ = Space$(buflen)+Chr$(0)
    CallDLL #wsock32, "recv", _
        s As Long, _
        Recv$ As Ptr, _
        buflen As Long, _
        flags As Long, _
        buflen As Long
    Recv$ = Left$(Recv$, buflen)
End Function

Function Send( s, buf$, flags )
    buflen=len(buf$)
    CallDLL #wsock32, "send", _
        s As Long, _
```

```
        buf$ As Ptr, _
        buflen As Long, _
        flags As Long, _
        Send As Long
End Function

'*** Winsock Thin Wrappers ***

Function accept( s )
    Struct p, length As Long
    p.length.struct = Len(sockaddr.struct)
    CallDLL #wsock32, "accept", _
        s As Long, _
        sockaddr As Struct, _
        p As Struct, _
        accept As Long
End Function

Function bind( s )
    namelen = Len(sockaddr.struct)
    CallDLL #wsock32, "bind", _
        s As Long, _
        sockaddr As Struct, _
        namelen As Long, _
        bind As Long
End Function

Function closesocket( s )
    CallDLL #wsock32, "closesocket", _
        s As Long, _
        closesocket As Long
End Function

Function htonl( hostlong )
    CallDLL #wsock32, "htonl", _
        hostlong As ULong, _
        htonl As ULong
End Function

Function htons( hostshort )
    CallDLL #wsock32, "htons", _
        hostshort As Word, _
        htons As Word
End Function

Function inetaddr( cp$ )
```

```
    CallDLL #wsock32, "inet_addr", _
        cp$ As Ptr, _
        inetaddr As ULong
End Function

Function listen( s, backlog )
    CallDLL #wsock32, "listen", _
        s As Long, _
        backlog As Long, _
        listen As Long
End Function


Function socket( af, type, protocol )
    CallDLL #wsock32, "socket", _
        af As Long, _
        type As Long, _
        protocol As Long, _
        socket As Long
End Function

Function WSAAsyncSelect( s, hWnd, wMsg, lEvent )
    CallDLL #wsock32, "WSAAsyncSelect", _
        s As Long, _
        hWnd As ULong, _
        wMsg As ULong, _
        lEvent As Long, _
        WSAAsyncSelect As Long
End Function

Sub WSACleanup
    CallDLL #wsock32, "WSACleanup", _
        r As Void
End Sub

Function WSAGetLastError()
    CallDLL #wsock32, "WSAGetLastError", _
        WSAGetLastError As Long
End Function

Function WSAStartup( wVersionRequested )
    CallDLL #wsock32, "WSAStartup", _
        wVersionRequested As Word, _
        WSAData As Struct, _
        WSAStartup As Long
End Function
```

```
Function connect( s )
    namelen = Len(sockaddr.struct)
    CallDLL #wsock32, "connect", _
        s As Long, _
        sockaddr As Struct, _
        namelen As Long, _
        connect As Long
End Function

'*** WMLiberty Thin Wrappers ***

Function SetWMHandler( hWnd, uMsg, lpfnCB, lSuccess )
    CallDLL #wmlib, "SetWMHandler", _
        hWnd As Long, _
        uMsg As Long, _
        lpfnCB As Long, _
        lSuccess As Long, _
        SetWMHandler As Long
End Function

function validip$(var$)
fail=0
    print var$
    for x=1 to len(var$)
        if mid$(var$,x,1)="." then var$=left$(var$,x-1)+" "+right$(var
$,len(var$)-x)
    next x
    print var$
    if word$(var$,5)<>"" then goto [endoffuncvalidip]
    if word$(var$,4)="" then goto [endoffuncvalidip]
    for x=1 to 4
        buf$=word$(var$,x)
        buf$=trim$(buf$)
        print buf$
        if len(buf$)>3 then fail=1
        if val(buf$)=0 and buf$<>"0" then fail=1
    next x
if fail=0 then validip$=var$
[endoffuncvalidip]
end function
```