ĆWICZENIE NR 5

Zadanie: zmodyfikować program tak aby można było połączyć trzech klientów oraz dodać do programu możliwość zdalnego uruchomienia kalkulatora.

Przykład: komunikator sieciowy oparty o gniazda sieciowe działający w trybie p2p. Aplikacja komunikuje się poprzez gniazda sieciowe wykorzystując protokół TCP/IP. Dwie główne procedury tx oraz rx służą odpowiednio do wysyłania danych i odbierania. Uruchamiane są jako wątki aby nie zawieszać działania całego programu. Program wykorzystuje bibliotekę ws2_32.lib którą należy zainicjować funkcją WSAStartup. Funkcja ta przyjmuje dwa parametry – numer wersji biblioteki adres struktury WSADATA w której zapisywane są informacje o gniazdach sieciowych.

```
INVOKE WSAStartup, 00000101h, OFFSET wsadata
```

Funkcja rx działa w trybie serwera. Aby go uruchomić musimy znać port na którym serwer będzie nasłuchiwał. W tym celu pobieramy go z pola edycji.

```
INVOKE GetDlgItemInt, hDlgEx, ID_PORTRX, ADDR Port, 0
mov Port, eax
```

Funkcją socket tworzy się gniazdo sieciowe. AF_INET oznacza typ protokołu TCP/IP a SOCK_STREAM komunikację połączeniową.

```
INVOKE socket, AF_INET, SOCK_STREAM, 0
```

`htons` konwertuje numer portu na format wykorzystywany w sieciach TCP/IP. Następnie uzupełniana jest struktura SockAddr wymagana do utworzenia połączenia.

```
INVOKE htons, Port

mov [SockAddr.sin_port],ax
mov [SockAddr.sin_family],AF_INET
mov [SockAddr.sin_addr],INADDR_ANY
```

`bind` powoduje podłączenie gniazda do medium komunikacyjnego. Pierwszy parametr jest deskryptorem gniazda, następnie struktura sockaddr_in oraz jej rozmiar. Po ustawieniu gniazda można zacząć nasłuchiwanie funkcją `listen` gdzie wymagany jest deskryptor gniazda oraz maksymalną długość kolejki wywołań.

```
INVOKE bind, hSocketRx, ADDR SockAddr, SIZEOF SockAddr
INVOKE listen, hSocketRx, 1
```

W pętli odbiera się wszystkie wiadomości przychodzące na określony port. `accept` ustanawia połączenie z klientem a `recv` zapisuje odebrane dane do bufora.

```
@p1:
      mov SockAddrLen, SIZEOF SockAddr
```

```
            INVOKE accept, hSocketRx, ADDR SockAddr, ADDR SockAddrLen
            mov hKlientRx, eax
            INVOKE recv, eax, OFFSET buffRx, 64, 0
            INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET txt2
            INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET buffRx
        jmp @p1
```

Funkcja tx poza portem adresata wymaga również jego adres sieciowy. Funkcją gethostbyaddr oraz gethostbyname zwracają wskaźnik na strukturę HOSTENT w której zapisane są informacje o hoście. Pierwsza z nich posiłkuje się nazwą sieciową komputera a druga adresem IP.

```
        INVOKE gethostname, ADDR szIP, SIZEOF szIP
        cmp eax, 0
        jne @e1

        INVOKE gethostbyname,ADDR szIP
        cmp eax, 0
        je @e1

        mov eax, dword ptr [eax+hostent.h_list]
        mov eax, [eax]
        mov eax, [eax]
        mov IP, eax
```

inet_ntoa konwertuje adres IP z zapisanego jako dword na zwykły napis. Następnie jest on wyświetlany w górnym polu edycji.

```
        INVOKE    inet_ntoa,DWORD PTR SockAddr.sin_addr
        INVOKE    lstrcpy,ADDR szIP,eax
        INVOKE    SetDlgItemText,hDlgEx,ID_IP,ADDR szIP
```

Pozostała część jest identyczna jak w rx poza funkcją `send` działającą odwrotnie do `recv`.

Kod przykładu:

```
include \masm32\include\masm32rt.inc
include \masm32\include\ws2_32.inc
includelib \masm32\lib\ws2_32.lib

ID_MAIN    = 100
ID_LISTBOX = 101
ID_WYSLIJ  = 102
ID_TEXT    = 103
ID_IP      = 104
ID_NAPIS   = 105
ID_NAPIS2  = 106
ID_PORTTX  = 107
ID_PORTRX  = 108
ID_NAPIS3  = 109
ID_START   = 110

.data?
hInstance dd ?
hLog      dd ?
hThread   dd ?
hDlgEx    dd ?
hKlientRx dd ?
```

```
hSocketRx dd ?
hSocketTx dd ?
wsadata WSADATA <>

.data
buffRx      db 64 dup(NULL)
buffTx      db 64 dup(NULL)
txt1 db ' :: listen',NULL
txt2 db ' :: rx',NULL
txt3 db ' :: tx',NULL
.code

   Rx proc
      LOCAL Port:DWORD
      LOCAL SockAddr:sockaddr_in
      LOCAL SockAddrLen:DWORD

      INVOKE GetDlgItemInt, hDlgEx, ID_PORTRX, ADDR Port, 0
        mov Port, eax

      INVOKE socket, AF_INET, SOCK_STREAM, 0
      mov hSocketRx, eax
      cmp eax, INVALID_SOCKET
      je @e1

      INVOKE htons, Port

      mov [SockAddr.sin_port],ax
      mov [SockAddr.sin_family],AF_INET
      mov [SockAddr.sin_addr],INADDR_ANY

      INVOKE bind, hSocketRx, ADDR SockAddr, SIZEOF SockAddr
      INVOKE listen, hSocketRx, 1
      cmp eax, 0
      jne @e1

      INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET txt1
      @p1:
             mov SockAddrLen, SIZEOF SockAddr
             INVOKE accept, hSocketRx, ADDR SockAddr, ADDR SockAddrLen
             mov hKlientRx, eax
             INVOKE recv, eax, OFFSET buffRx, 64, 0
             INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET txt2
             INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET buffRx
      jmp @p1

      @e1:
      ret
   Rx endp

  tx proc parametr:DWORD
     LOCAL SockAddr:sockaddr_in
     LOCAL IP:DWORD
     LOCAL szIP[128]:BYTE
     LOCAL Port:DWORD

     INVOKE    GetDlgItemText,hDlgEx,ID_IP,ADDR szIP,SIZEOF szIP

     INVOKE GetDlgItemInt, hDlgEx, ID_PORTTX, ADDR Port, 0
       mov Port, eax

     INVOKE gethostbyname,ADDR szIP
     cmp eax, 0
     jne AddrName

     INVOKE    inet_addr,ADDR szIP
     mov IP,eax
```

```
  INVOKE    gethostbyaddr,ADDR IP,4,AF_INET

  AddrName:

  cld
  lea esi,dword ptr [eax+hostent.h_len]
  lea edi,SockAddr.sin_addr
  lodsw
  movzx ecx,ax
  lodsd
  mov esi,dword ptr [eax]
  rep movsb


  INVOKE    inet_ntoa,DWORD PTR SockAddr.sin_addr

  INVOKE    lstrcpy,ADDR szIP,eax

  INVOKE    SetDlgItemText,hDlgEx,ID_IP,ADDR szIP


  INVOKE    htons,Port
  mov    [SockAddr.sin_port],ax
  mov    [SockAddr.sin_family],AF_INET

  INVOKE    socket,AF_INET,SOCK_STREAM,0
  mov    hSocketTx,eax

  INVOKE    connect,hSocketTx,ADDR SockAddr,SIZEOF SockAddr


  INVOKE GetDlgItemText,hDlgEx,ID_TEXT,ADDR buffTx,SIZEOF buffTx
  INVOKE lstrlen, OFFSET buffTx
  INVOKE send, hSocketTx, OFFSET buffTx, 64, 0
    INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET txt3
    INVOKE SendMessage, hLog, LB_ADDSTRING, 0, OFFSET buffTx
  .IF hSocketTx != 0
          INVOKE closesocket, hSocketTx
          mov hSocketTx, 0
  .ENDIF
  ret
tx endp

DlgProc proc hDlg,uMsg,wParam,lParam:DWORD
  pushad
  .IF uMsg==WM_INITDIALOG

          INVOKE GetDlgItem,hDlg,ID_LISTBOX
          mov hLog,eax

          mov eax, hDlg
          mov hDlgEx, eax

  .ELSEIF  uMsg==WM_CLOSE
          .IF hSocketRx != 0
                INVOKE closesocket, hSocketRx
          .ENDIF

          .IF hKlientRx != 0
                INVOKE closesocket, hKlientRx
          .ENDIF

        INVOKE TerminateThread, hThread, 0
     INVOKE EndDialog,hDlg,0
  .ELSEIF uMsg==WM_COMMAND
      .IF wParam == ID_WYSLIJ
```

```
            INVOKE CreateThread, NULL, 0, OFFSET tx, 0, 0, OFFSET hThread
            .ELSEIF wParam == ID_START
                INVOKE CreateThread, NULL, 0, OFFSET Rx, 0,0,OFFSET hThread
        .ENDIF
    .ENDIF

    popad
    xor eax,eax
    ret
DlgProc endp

Start:

    INVOKE GetModuleHandle,NULL
    mov hInstance,eax

    INVOKE WSAStartup, 00000101h, OFFSET wsadata
    INVOKE DialogBoxParam,hInstance,ID_MAIN,0,ADDR DlgProc,0

    INVOKE ExitProcess,0

END Start
```

Plik .rc przykładu:

```
#include "..\include\resource.h"

#define ID_MAIN    100
#define ID_LISTBOX 101
#define ID_WYSLIJ  102
#define ID_TEXT    103
#define ID_IP      104
#define ID_NAPIS   105
#define ID_NAPIS2  106
#define ID_PORTTX  107
#define ID_PORTRX  108
#define ID_NAPIS3  109
#define ID_START   110


ID_MAIN DIALOG 0, 0, 214, 210
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Klient"
FONT 8, "Ms Shell Dlg"
{

    LISTBOX         ID_LISTBOX, 4, 32, 205, 137, WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "Wyslij", ID_WYSLIJ, 157, 190, 47, 14
    EDITTEXT        ID_TEXT, 4, 190, 148, 13, ES_AUTOHSCROLL
    EDITTEXT        ID_IP, 24, 12, 53, 12, ES_AUTOHSCROLL
    LTEXT           "IP", ID_NAPIS, 9, 15, 8, 8, SS_LEFT
      LTEXT           "PORT_tx", ID_NAPIS2, 80, 15, 30, 8, SS_LEFT
      EDITTEXT        ID_PORTTX, 110, 12, 30, 12, ES_AUTOHSCROLL
      LTEXT           "PORT_rx", ID_NAPIS3, 145, 15, 30, 8, SS_LEFT
      EDITTEXT        ID_PORTRX, 180, 12, 30, 12, ES_AUTOHSCROLL
      PUSHBUTTON      "Start", ID_START, 157, 170, 47, 14
}
```