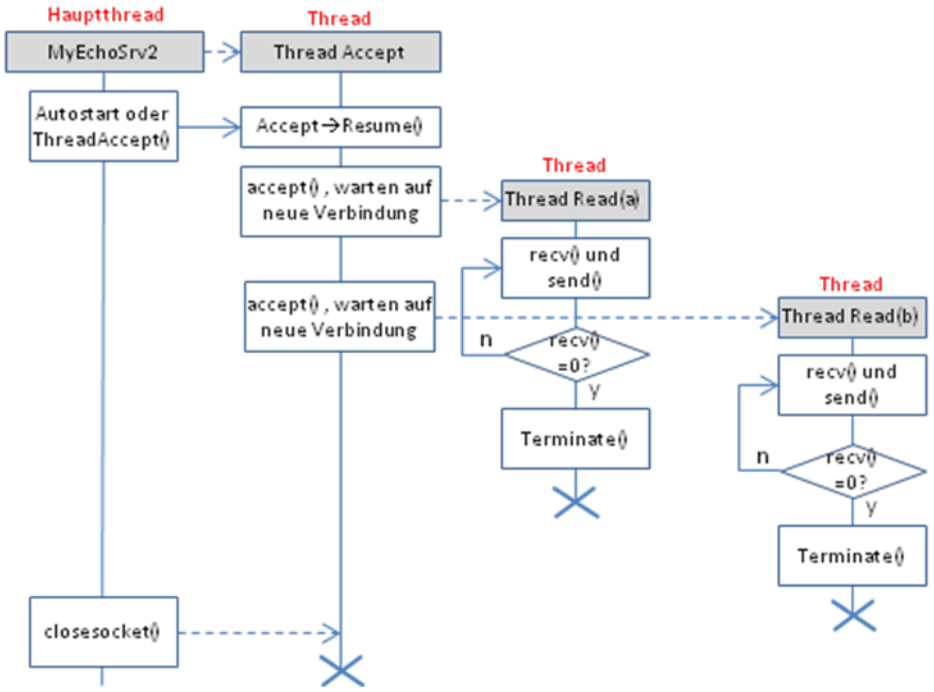
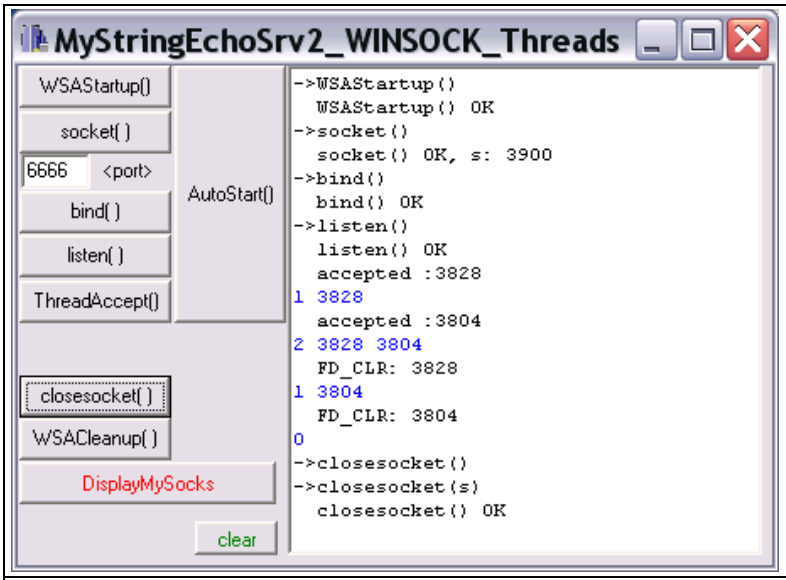
1.1 Ziel des Projektes

- Es soll ein String-Echo-Server auf WINSOCK-API aufgesetzt werden.
- Die blockierenden Funktion:
  - accept() soll in einem Thread Accept aufgerufen werden. Dieser Thread wird beim Start des Servers erzeugt und zum Ablauf gebracht. Für jede angenommene Verbindung wird dynamisch ein neuer Thread Receive erzeugt.
  - recv() soll in einem Thread Receive aufgerufen werden. Wird der Socket geschlossen, terminiert sich der Thread Receive selber.



1.2 Realisierung des Projektes MyStringEchoSrv2_WINSOCK_Threads



- Erzeugen Sie ein neues Projekt "MyStringEchoSrv2_WINSOCK_Threads" im gleichnamigen Order.
- Editieren Sie die Oberfläche entsprechend der nebenstehenden Abbildung.
- Erzeugen Sie Schritt für Schritt den Programmcode, entsprechend dem Beispiel-Code.

Autostart ruft die Funktionen WSAStartup(), socket(), bind(), listen(), Accept→Resume() auf.

In dem Beispiel sieht man folgendes:

- AutoStart() wurde aufgerufen: ein ServerSocket mit dem Descriptor 3900 wurde errichtet, der auf ankommende Echo-Clientverbindungen wartet.
- Es werde zwei Verbindungen von EchoClients angenommen mit den Descriptoren 3828, 3804.
- Die EchoClients beenden die Verbindung. Die Descriptoren werden aus der fd_set mySocks entfernt

Unit1.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <winsock2.h> //erforderlich
#include <string>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
TThread *Accept =  new TAccept(True); //erzeugen, und nicht starten
u_int s;                      //socketDescriptor Server
int ret_len;                  //für send() und recv()
char recv_buf[4095];    //recv()
struct sockaddr_in xAddr; int xAddrLen; //für bind(),accept()
fd_set mySocks;         //array für sockets aller akzeptierten verbindungen
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
            : TForm(Owner)
{

}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void __fastcall TForm1::WSAStartup_Click(TObject *Sender)
{
 WSADATA wsaD;
//an DLL anmelden
  RichEdit1->Lines->Add("->WSAStartup()");
  if  (WSAStartup (MAKEWORD(2,0),&wsaD) ==0) //MAKEWORD(maj,min)
        RichEdit1->Lines->Add("  WSAStartup() OK");
  else RichEdit1->Lines->Add("  WSAStartup() ERR");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::socket_Click(TObject *Sender)
{
  RichEdit1->Lines->Add("->socket()");
  s = socket(AF_INET,SOCK_STREAM,0);
  if (s!=0){RichEdit1->Lines->Add("  socket() OK, s: "+IntToStr(s)); }
  else      RichEdit1->Lines->Add("  socket() ERR");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::bind_Click(TObject *Sender)
{
   xAddr.sin_family=AF_INET;
   xAddr.sin_port=htons(port->Text.ToInt());
   xAddr.sin_addr.S_un.S_addr=inet_addr("0.0.0.0");
   RichEdit1->Lines->Add("->bind()");
   if (bind (s,(struct sockaddr*)&xAddr,sizeof(sockaddr))==0)
            RichEdit1->Lines->Add("  bind() OK");
   else RichEdit1->Lines->Add("->connect() ERR: "+IntToStr(WSAGetLastError()));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::listen_Click(TObject *Sender)
{
   RichEdit1->Lines->Add("->listen()");
   if (listen(s,5)==0) RichEdit1->Lines->Add("  listen() OK");
   else RichEdit1->Lines->Add("  listen() ERR: "+IntToStr(WSAGetLastError()));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::closesocket_Click(TObject *Sender)
{  u_int x;
   RichEdit1->Lines->Add("->closesocket()");
//--erst alle client-sockets schließen
   while (mySocks.fd_count > 0)
```

```
      {
        x=mySocks.fd_array[0];
        if (closesocket(x)!=SOCKET_ERROR)
             {FD_CLR(x,&mySocks);//eintrag löschen
              RichEdit1->Lines->Add("  closesocket() OK");}
        else RichEdit1->Lines->Add("  closesocket() ERR:
                                   "+IntToStr(WSAGetLastError()));
      }//end-while (mySocks...
//Thread Accept terminieren. ACHTUNG: Anwendung muss dann neu gestartet werden
   Accept->FreeOnTerminate; Accept->Terminate();
//ServerSocket schließen
   RichEdit1->Lines->Add("->closesocket(s)");
   if (closesocket(s)!=SOCKET_ERROR)
    {FD_CLR(x,&mySocks);//eintrag löschen
     RichEdit1->Lines->Add("  closesocket() OK");}
   else RichEdit1->Lines->Add("  closesocket()ERR:  "+IntToStr(WSAGetLastError()));}
//---------------------------------------------------------------------------
void __fastcall TForm1::WSACleanup_Click(TObject *Sender)
{
   RichEdit1->Lines->Add("->WSACleanup()");
   if (WSACleanup()==0) RichEdit1->Lines->Add("  WSACleanup() OK");
   else RichEdit1->Lines->Add("  WSACleaup() ERR:"+IntToStr(WSAGetLastError()));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::clear_Click(TObject *Sender)
{
   RichEdit1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::automatisch_Click(TObject *Sender)
{
   WSAStartup_Click(Form1);
   socket_Click(Form1);
   bind_Click(Form1);
   listen_Click(Form1);
   Accept->Resume();//Thread starten
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Zeige_mySocks_Click(TObject *Sender)
{
   RichEdit1->SelAttributes->Color=clBlue;
   AnsiString txt;
   txt=IntToStr(mySocks.fd_count);
   if (mySocks.fd_count!=0)
    for (u_int i=0;i<mySocks.fd_count;i++ )
         txt=txt+" "+IntToStr(mySocks.fd_array[i]);}
   RichEdit1->Lines->Add(txt);
   RichEdit1->SelAttributes->Color=clBlack;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::accept_Click(TObject *Sender)
{
   Accept->Resume();//Thread starten
}
//---------------------------------------------------------------------------
```

1.3 Der Thread für Accept

Aus Borland Entwickler-Handbuch, Kapitel 11, Multithread-Anwendungen implementieren:
„Um ein Thread-Objekt in einer Anwendung zu implementieren, erzeugen Sie einen neuen Nachkommen von *TThread*. Wählen Sie dazu zunächst im Hauptmenü den Befehl ***Datei / Neu / Andere***. Doppelklicken Sie anschließend im Dialogfeld ***Objektgalerie*** auf das Symbol ***Thread-Objekt*** und vergebem Sie einen Klassennamen für das neue Thread-Objekt, z. B. *TMeinThread*. Wenn Sie auf *OK* klicken, erstellt C++Builder eine neue CPP-Datei sowie die Header-Datei, um den *Thread* zu implementieren.“

Unit2.cpp → Thread TAccept

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <winsock2.h> //erforderlich
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#pragma package(smart_init)
extern fd_set mySocks;
extern int s;
//---------------------------------------------------------------------------
__fastcall TAccept::TAccept(bool CreateSuspended)
      : TThread(CreateSuspended)
{
}
//---------------------------------------------------------------------------
void __fastcall TAccept::Execute()
{
//---- Thread-Code hier platzieren ----
      while(!Terminated)
      {
       as=accept(s,0,0);
       if (as>0)
         {
            Form1->RichEdit1->Lines->Add("  accepted :"+ IntToStr(as));
//sockDescriptors in Struktur mySocks speichern, damit sie ggf. geschlossen werden
            FD_SET(as,&mySocks);
//Thread erzeugen,starten und sockDescriptor übergeben
            TThread *Read =  new TRead(False,as);
            Form1->Zeige_mySocks_Click(Form1);
         }
       }
}
//---------------------------------------------------------------------------
```

Unit2.h → Thread TAccept

```
//---------------------------------------------------------------------------
#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#include <Classes.hpp>
//---------------------------------------------------------------------------
class TAccept : public TThread
{
private:
protected:
      void __fastcall Execute();
      int as; //locale Variable für sockesDescriptor der akzeptierten Assoziation
public:
      __fastcall TAccept(bool CreateSuspended);
};
//---------------------------------------------------------------------------
#endif
```

1.4 Der Thread für Read

Legen Sie ein weitere Thred-Klasse TRead an (siehe 1.3).

Unit3.cpp → Thread TRead

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <winsock2.h>
#include "Unit3.h"
#include "Unit1.h"
#pragma package(smart_init)
```

```
extern fd_set mySocks;
//---------------------------------------------------------------------------
__fastcall TRead::TRead(bool CreateSuspended, int readSockDescriptor)
      : TThread(CreateSuspended)
{
   readSock=readSockDescriptor;
}
//---------------------------------------------------------------------------
void __fastcall TRead::Execute()
{
      while(!Terminated)
      {
       readLen = recv(readSock, readBuf, sizeof(readBuf),0);
       switch (readLen)
            {
         case -1 : ;break;
         case 0  : {Form1->RichEdit1->Lines->Add("  FD_CLR: "+IntToStr(readSock));
                        FD_CLR(readSock,&mySocks);
                        Form1->Zeige_mySocks_Click(Form1);
                        this->FreeOnTerminate=True;
                        this->Terminate();
                        break; }
         default : {AnsiString text=((AnsiString)readBuf).SubString(0,readLen);
                        sendLen=send(readSock,readBuf,readLen,0); }
            }//end-switch
      }//end-while
}
//---------------------------------------------------------------------------
```

Unit3.h → Thread TRead

```
//---------------------------------------------------------------------------
#ifndef Unit3H
#define Unit3H
//---------------------------------------------------------------------------
#include <Classes.hpp>
//---------------------------------------------------------------------------
class TRead : public TThread
{
private:
protected:
      void __fastcall Execute();
//lokale Variable jeder Thread-Instanz
      char readBuf[4095];
      fd_set readfds;
      int readSock, readLen, sendLen;
public:
      __fastcall TRead(bool CreateSuspended, int readSockDescriptor);
};
//---------------------------------------------------------------------------
#endif
```