| MyTimeClient-TCP | | Prof. Dr.-Ing. habil. L. Winkler | HOCHSCHULE MITTWEIDA UNIVERSITY OF APPLIED SCIENCES |
|---|---|---|---|
| CPP-Project mit TClientSocket | 2015-01 | Fakultät Informations- und Elektrotechnik | |

# 1 Ziel des Projektes

- Nutzung eines TCP-Client-Socket der Klasse TClientSocket[1] aus der RAD-Studio-IDE.
- Es soll eine Client-Anwendung programmiert werden, die von einem dem Zeitserver **time-d.nist.gov** (RFC 868 „Time Protocol“) die Zeit abfragt und diese in lesbarer Form darstellt. Sollte dieser Server passiv sein, findet man Alternativen unter: http://tf.nist.gov/tf-cgi/servers.cgi
- Verschaffen Sie sich zunächst anhand des RFC 868 einen Überblick, wie dieser Server arbeitet.
- Verschaffen Sie sich mittels der Hilfefunktion unter RAD-Studio-XE einen Überblick zu den Zeit/Datumsfunktionen **localtime()**, **asctime()**

# 2 Grundlagen

Ein nach RFC 868 arbeitender Zeitserver, liefert die vergangenen Sekunden seit dem 1.1.1900. Er stellt diese in 4 Bytes (LongWord) in der so genannten Network-Order (big-endian) zur Verfügung.
Damit kann der Server bis **$2^{32}$ = 4.294.967.296** zählen.

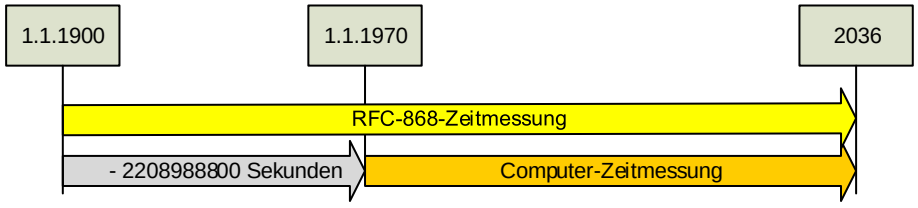
✎ Wie viel Jahre kann man diesen Zeitserver nutzen?

$$Anzahl_{Jahre} = \frac{4294967296}{s*m*h*d} = \frac{4294967296}{60*60*24*365} = \frac{4294967296}{3024000} \approx \underline{\underline{136{,}19}}$$

Dieser Zeitserver ist damit rund von 1900 bis zum Jahr 2036 nutzbar.

Viele Funktionen zur Datumsberechnung in Computern basieren aber auf einer Zählung der Sekunden ab 1.1.1970. Deshalb muss man, wenn man Computer-Datumsfunktionen nutzen will, diesen Zeitstempel auf den 1.1.1970 normalisieren. Von dem Zeitstempel des Time-Servers muss man deshalb die Sekunden für **70 Jahre a 365 Tage + 17 Tage dazu wegen der Schaltjahre** abziehen:

s*m*h*365*70 + s*m*h*17 = 60*60*24*365*70 + 60*60*24*17 = **2208988800**



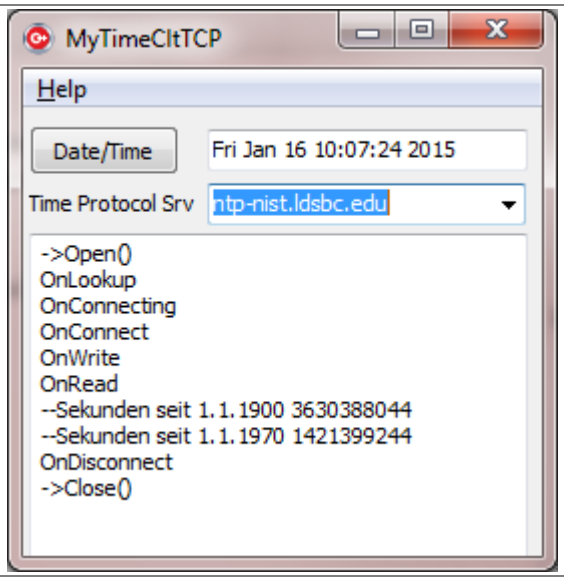

**Auszug aus RFC 868 Time Protocol**

```
This protocol may be used either above the Transmission Control Protocol (TCP) or
above the User Datagram Protocol (UDP).
When used via TCP the time service works as follows:
   S: Listen on port 37 (45 octal).
   U: Connect to port 37.
   S: Send the time as a 32 bit binary number.
   U: Receive the time.
   U: Close the connection.
   S: Close the connection.
 The server listens for a connection on port 37.  When the connection   is
established, the server returns a 32-bit time value and closes the  connection.  If
the server is unable to determine the time at its  site, it should either refuse the
connection or close it without sending anything.
```

[1] Die Komponente ist nicht standardmäßig in die IDE eingebunden. Zum Einbinden der Komponente geht man wie folgt vor:
(1) Im Menü: Komponenten→Packages installieren
(2)→Hinzufügen
(3) C|D|X:\Programme\Embarcardero\RAD Studio\9.0\bin\dclockets160bpl wählen →Öffnen
(4)Die Komponentenpalette „Internet“ hat jetzt zwei neue Komponenten: TClientSocket, TServerSocket

| **Datum- und Zeitfunktionen** | |
|---|---|
| Header-Datei **time.h** einbinden, für Datentypen und Funktionen | `#include <time.h>` |
| Damit sind die Datentypen **time_t** | LongWord, 4-Byte-Wert |
| und **tm** nutzbar | Struktur mit folgendem Aufbau:<br>`struct tm {`<br>`  int tm_sec;  // Sekunden (0..59)`<br>`  int tm_min;  // Minuten (0..59)`<br>`  int tm_hour;  // Stunden (0..23)`<br>`  int tm_mday;  // Tag des Monats (1..31)`<br>`  int tm_mon;  // Monat im Jahr (0..11)`<br>`  int tm_year;  // Jahr seit 1900`<br>`  int tm_wday;  // Tage seit Sonntag (Wochentag) (0..6)`<br>`  int tm_yday;  // Tage seit Neujahr (1.1.) (0..365)`<br>`  int tm_isdst;  // Sommerzeit-Flag`<br>`}` |
| Funktion **time()**, zur Ermittlung des Zeitstempels von der PC-Uhr | `//Ermittlung des Zeitabstandes vom 1.1.1970 in Sekunden`<br>`time_t t;  //in t wird der Zeitstempel gespeichert`<br>`t=time(NULL);  //in t stehen die Sekunden ab 1.1.1970` |
| Ausgabe der Sekunden in Edit1 | `Edit1->Text= (AnsiString)((LongWord)t);` |
| Wir holen den Zeitstempel aber von einem “Time Protocol”-Server | `ClientSocket1->Socket->ReceiveBuf(&t,4);`<br>`t=(ntohl(t)); //network order in host order wandeln`<br>`//In t steht der Zeitstempel bezogen auf 1.1.1900. Nach`<br>`t=t-2208988800;`<br>`//steht in t der Zeitstempel bezogen auf 1.1.1970` |
| Funktion **localtime()**, konvertiert aus einer Variablen vom Typ **time_t** Datum und Uhrzeit in eine Struktur vom Typ **tm**. | `struct tm *ts;  //Pointer auf die Variable ts vom Typ tm`<br>`ts=localtime(&t); /*localtime() gibt einen Zeiger auf eine Struktur vom Typ tm zurück, wo die aktuelle Zeit und das Datum stehen (siehe oben)*/` |
| Die Funktion **asctime()** konvertiert aus **ts** Datum u. Uhrzeit in eine Zeichenkette (26 characters) der Form:<br>**Mon Nov 21 11:31:54 1983\n\0**<br>Diese wird in Edit1 ausgegeben. | `Edit1->Text=((AnsiString)asctime(ts)).SubString(0,24);`<br><br>`\n ->carriage return`<br>`\0 ->markiert in C das Ende einer Zeichenkette` |

## 3 Realisierung des Projektes MyTime_TClientSocket




- Erzeugen Sie ein neues Projekt “MyTimeCltTCP“ im gleichnamigen Order.
- Nutzen Sie die Komponente TClientSocket aus der Palette „Internet“
  - Editieren Sie die Oberfläche entsprechend der nebenstehenden Abbildung. Nutzen Sie:
  - TMenu,
  - TButton, Button1
  - TEdit, Edit1
  - TLabel, Label1
  - TComboBox, ComboBox1
  - TMemo, Memo1
- Erzeugen Sie Schritt für Schritt den Programmcode, entsprechend dem Beispiel-Code.

Grün: Header und Kommentare
Rot: Socketfunktionen
Blau: Zeit-/Datumsfunktionen

```
//==notwendiger Programmtext==

#include <vcl.h>
#pragma hdrstop
#include <time.h>   //erforderlich
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
time_t t;       //t ist LongWord zur Speicherung des Zeitstempels
struct tm *ts;  //*ts ist Pointer auf Struktur für Zeit-/Datumsparameter

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void __fastcall TForm1::Info1Click(TObject *Sender)
{
      ShowMessage("Wenn Server passiv, anderen wählen. Siehe:
      http://tf.nist.gov/tf-cgi/servers.cgi" );
}

//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
      Memo1->Clear();
      ClientSocket1->Host=ComboBox1->Text;
      Memo1->Lines->Add("->Open()");
      ClientSocket1->Open();
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Read(TObject *Sender,
       TCustomWinSocket *Socket)
{
      ClientSocket1->Socket->ReceiveBuf(&t,4);
      if (t>0){
      t= (ntohl(t)); //Zeitstempel in die Hostorder konvertieren
      Memo1->Lines->Add("Sek. Seit 1.1.1900"+(AnsiString)((LongWord)(t)));
      t=t-2208988800;
      Memo1->Lines->Add("Sekunden seit 1.1.1970 "+(AnsiString)t);
//--die normalisierte zeit mittels Standardfunktionen ausgeben
      /* Datum und Zeit in eine Struktur konvertieren */
      ts = localtime(&t);
      /* Datum und Zeit ausgeben */
      Edit1->Text=((AnsiString)asctime(ts)).SubString(0,24);
      }else Memo1->Lines->Add("—keinen Zeitstempel erhalten");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Disconnect(TObject *Sender,
        TCustomWinSocket *Socket)
{
        Memo1->Lines->Add("^^OnDisconnect");
       Memo1->Lines->Add("->Close()");
        ClientSocket1->Close();
}
//---------------------------------------------------------------------------
```

```
//==nicht notwendiger Programmtext, dient nur zur Ereignisanzeige
//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Lookup(TObject *Sender,
      TCustomWinSocket *Socket)
{
       Memo1->Lines->Add("^^OnLookup");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Connecting(TObject *Sender,
     TCustomWinSocket *Socket)
{
      Memo1->Lines->Add("^^OnConnecting");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Connect(TObject *Sender,
       TCustomWinSocket *Socket)
{
      Memo1->Lines->Add("^^OnConnect with "+ClientSocket1->Address);
}
//---------------------------------------------------------------------------
```