1.1 Ziel des Projektes

- Grundlegender Umgang mit einem Client-Socket der Klasse TClientSocket[1] aus der Borland-IDE.
- Es soll eine Client-Anwendung programmiert werde, die von einem Zeitserver (141.55.192.51) die Zeit abfragt und diese in lesbarer Form darstellt.
- Der Zeitserver arbeitet entsprechend RFC 868.
- Verschaffen Sie sich zunächst anhand des RFC einen Überblick, wie dieser Server arbeitet.
- Verschaffen Sie sich mittels der Hilfefunktion unter Borland-CPP-Builder einen Überblick zu den Zeit/Datumsfunktionen asctime(), localtime()

1.2 Grundlagen

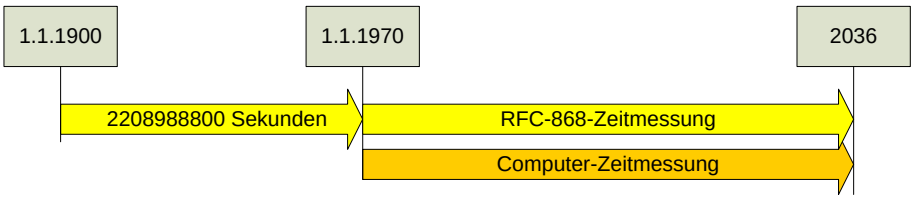
Ein nach RFC 868 arbeitender Zeitserver, liefert die vergangenen Sekunden seit dem 1.1.1900. Er stellt diese in 4 Bytes (LongWord) in der so genannten Network-Order (big-endian) zur Verfügung.
Damit kann der Server bis $\mathbf{2^{32} = 4.294.967.296}$ zählen.
✎ Wie viel Jahre kann man diesen Zeitserver 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 2208988800 Sekunden abziehen (70 Jahre a 365 Tage + 17 Tage dazu wegen der Schaltjahre):

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.
```

[1] Die Komponente ist nicht standardmäßig in die IDE eingebunden. Zum Einbinden der Komponente geht man wie folgt vor:
(1) In BDS2006 Menü: Komponenten→Packages installieren (2)→Hinzufügen
(3) C|D:\Programme\Borland\BDS\4.0\Bin\dclsockets100bpl wählen →Öffnen (4)Die Komponentenpalette „Internet“ hat jetzt zwei neue Komponenten: TClientSocket, TServerSocket

```
   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.
```

localtime

Header-Datei `<time.h >`
Kategorie Uhrzeit- und Datumsroutinen
Prototyp `struct tm *localtime(const time_t *timer);`
Beschreibung Konvertiert Datum und Zeit in eine Struktur. *localtime* akzeptiert die Adresse eines von *time* zurückgegebenen Werts und gibt einen Zeiger auf eine Struktur des Typs *tm* zurück, die Zeitwerte enthält. Die Funktion berücksichtigt die Zeitzone und gegebenenfalls die Sommerzeit.

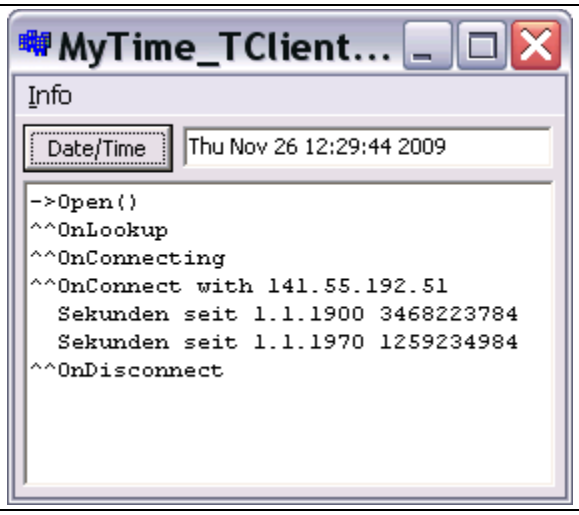
Die Struktur *tm* ist in der Header-Datei time.h folgendermaßen deklariert:

```
struct tm { int tm_sec; int tm_min; int tm_hour; int tm_mday; int tm_mon; int tm_year;
int tm_wday; int tm_yday; int tm_isdst; };
```

asctime

Header-Datei `<time.h >`
Kategorie Uhrzeit- und Datumsroutinen
Prototyp `char *asctime(const struct tm *tptr);`
The function stores in the static-duration time string a 26-character English-language representation of the time encoded in *tptr. It returns the address of the static-duration time string. The text representation takes the form:
`Sun Dec  2 06:55:15 1979\n\0`

1.3 Realisierung des Projektes MyTime_TClientSocket



- Erzeugen Sie ein neues Projekt “MyTime_TClientSocket“ im gleichnamigen Order.
- Nutzen Sie die Komponente TClientSocket aus der Palette „Internet“
- Editieren Sie die Oberfläche entsprechend der nebenstehenden Abbildung.
- Erzeugen Sie Schritt für Schritt den Programmcode, entsprechend dem Beispiel-Code.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <winsock.h>//erforderlich
#include <time.h>   //erforderlich
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
time_t zeit;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
```

```
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Info1Click(TObject *Sender)
{
      ShowMessage("Datum und Uhrzeit vom Time-Server\r194.94.124.1 Port 37" );
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
      Memo1->Clear();
      Memo1->Lines->Add("->Open()");
      ClientSocket1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Read(TObject *Sender,
      TCustomWinSocket *Socket)
{
       ClientSocket1->Socket->ReceiveBuf(&zeit,4);
       if (zeit>0)
       {
        zeit= (ntohl(zeit));
        Memo1->Lines->Add("Sekunden seit 1.1.1900 "+(AnsiString)((LongWord)(zeit)));
        zeit=zeit-2208988800;
        Memo1->Lines->Add("Sekunden seit 1.1.1970 "+(AnsiString)zeit);
        //die normalisierte zeit mittels Standardfunktionen ausgeben
        struct tm *tblock;
        /* Datum und Zeit in eine Struktur konvertieren */
        tblock = localtime(&zeit);
        /* Datum und Zeit ausgeben */
        Edit1->Text=((AnsiString)asctime(tblock)).SubString(0,24);
       }
}
//---------------------------------------------------------------------------
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);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ClientSocket1Disconnect(TObject *Sender,
        TCustomWinSocket *Socket)
{
        Memo1->Lines->Add("^^OnDisconnect");
}
//---------------------------------------------------------------------------
```