1. Betriebssystemübungen mit ObjectPascal

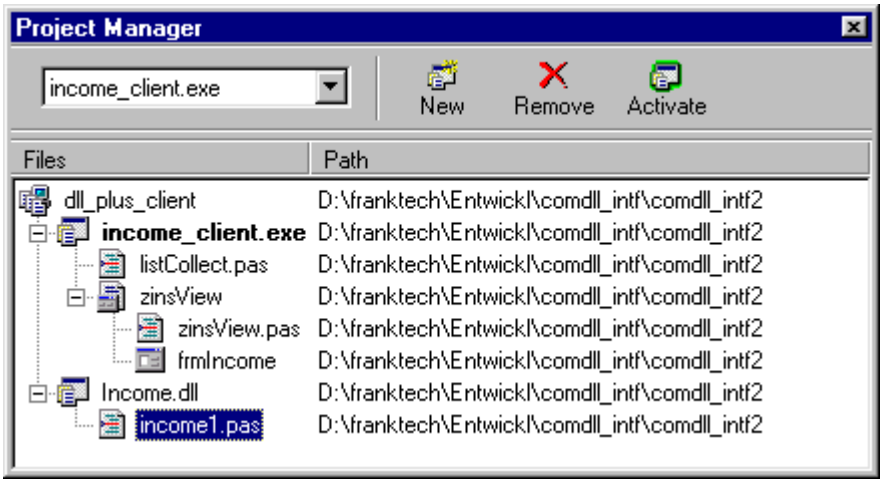
Im Rahmen dieser Übungen vertiefen wir unsere Kenntnisse in den 5 Domänen eines OS (Operating System), wobei die eigentliche GUI ja auch eine Domäne im Schichtenmodell eines OS darstellt:

- Prozesse
- Speicherverwaltung
- Dateisystem
- Peripherie
- Vernetzung

Wir arbeiten immer mit dem gleichen Client als Form (income_client.exe) in Delphi, Kylix oder FreePascal, der Schritt für Schritt an Funktionalität gewinnt (siehe Aufgabe 18 und Schlussdemo).
Einige Übungen benötigen das Buch „Patterns konkret“ von Max Kleiner. [i]
Auch das komplette Projekt mit Code ist verfügbar.[ii]

2. Aktivieren der DLL mit einem Client (Exe)

Zuerst richten wir uns die Entwicklungsumgebung so ein, dass ein hin- und herschalten mit entsprechendem Compilieren und Testen zwischen DLL und Client möglich ist. Starten Sie das Projekt innerhalb der Entwicklungsumgebung. Nachdem Sie die Benutzeroberfläche der Anwendung im Formular entworfen und den erforderlichen Quelltext geschrieben haben, können Sie das Projekt in der IDE oder von der Befehlszeile aus compilieren. Sie können in der Projektverwaltung den Inhalt von Projekten und Gruppen ansehen. Dort werden die Namen der Units, die in einer Unit enthaltenen Formulare (sofern vorhanden) und die Pfade der Projektdateien in einer hierarchischen Ansicht angezeigt. Unser Projekt stellt sich als Client und DLL wie folgt dar:



3. Kompilieren einer DLL

Unter einer dynamisch ladbaren Bibliothek versteht man in Windows eine dynamische Linkbibliothek (DLL) und in Linux eine Bibliotheksdatei mit gemeinsamen Objekten. Es handelt sich dabei um eine Sammlung von Routinen, die von Anwendungen und von anderen DLLs bzw. gemeinsamen Objekten aufgerufen werden können.
Dynamisch ladbare Bibliotheken enthalten wie Units gemeinsam genutzten Code und Ressourcen. Sie stellen jedoch eine separat kompilierte, ausführbare Datei dar, die zur Laufzeit zu den Programmen gelinkt wird, die sie verwenden.
DLLs sind Module aus kompiliertem Quelltext, die erst zusammen mit einer ausführbaren Datei Funktionalität für eine Anwendung liefern.
Aufgabe: Compilieren Sie zuerst die DLL (income.dll), dann die zugehörige EXE.

4. Aus welchen Kernel-DLL's besteht Windows

Öffnen Sie die bestehende Income-DLL mit der Schnellansicht (QuickView) oder einem anderen Viewer und notieren sich die referenzierten DLL's, die Sie mit der Schnellansicht im Abschnitt der „Import Table“ finden. Unter Linux werden jedoch DLLs (und Packages) als gemeinsam genutzte Objekte neu kompiliert. Welches sind gemäss der Häufigkeit die wichtigsten 4 DLLs und wozu dienen einige Funktionen dem Namen nach?

5. Wie wird eine DLL vom OS geladen

Wird ein Modul (Unit oder Bibliothek) in den Adressraum eines Prozesses geladen, versucht Windows, das Modul an dessen Standard-Image-Basisadresse abzulegen. Sollte das nicht möglich sein, weil die vorgegebene Adresse bereits belegt ist, wird das Modul an einer Adresse abgelegt, die Windows während der Laufzeit bestimmt (Adressverschiebung). Eröffnen Sie im Verzeichnis der DLL ein neues Projekt und implementieren den Zugriff auf die DLL. Wie weiss der OS-Loader, welche Adresse die DLL hat? Achten Sie auf den individuellen Entry-Point beim Aufruf. S. 361ff

6. Adressieren von Speicher

Ein Zeiger ist eine Variable, die eine Speicheradresse angibt. Wenn ein Zeiger die Adresse einer anderen Variablen enthält, zeigt er auf die Position dieser Variablen im Speicher oder auf die Daten, die an dieser Position gespeichert sind. Im Buch Patterns konkret auf Seite 29 gehen wir genauer auf die Speicherverwaltung mit Zeigern ein. Wie viele Seiten (Paging) ergibt eine 32-Bit Adressierung, wenn die Seite 4Kbyte gross ist.
Testen Sie die Speicheradressen mit dem Debugger anhand der Abbildungen 1.11, 1.12 und 1.13 auf Seite 30ff mit dem vorgängig erstellten DLL-Client.

7. Versenden einer Windows-Nachricht

Ab Seite 41 analysieren wir mal, wie ein OS auf all die Ereignisse reagieren kann und lernen so denn Event-Loop kennen, der fast allen Betriebssystemen mit einer GUI gemeinsam ist.
Ein Ereignis ist ein Mechanismus, der einen Vorgang mit einem Teil des Programmcodes verknüpft. Genauer gesagt handelt es sich bei einem Ereignis um einen Methodenzeiger, der auf eine Methode in einer bestimmten Klasseninstanz zeigt.
Suchen Sie in unserem Projekt einen Ereignishandler und beschreiben, wie der funktioniert.

8. Ereignisse an den Administrator

Kehren wir kurz der Technik den Rücken und lösen eine technisch-organisatorische Frage. Im Buch auf Seite 138 (Testing) und 250 (Parser) befindet sich jeweils ein Screenshot. Erstellen Sie eine Liste, welche wichtigen Ereignisse eines Systems das OS dem Administrator benachrichtigen sollen.

9. Threads, die schnellen Prozessfäden

Steigen wir in die Welt der Threads ein, dazu öffnen Sie das Demo unter Teil_3\Master_Slave. Was sind Threads und wie funktionieren die schnellen Dinger? (siehe auch S. 370)
Das Beispiel erzeugt einen Nachkommen von TThread, um einen Ausführungs-Thread in einer mehrschichtigen Anwendung darzustellen. Jede weitere Instanz eines TThread-Nachkommen bildet einen neuen Ablaufstrang. Wenn eine Anwendung über mehrere solcher Threads verfügt, wird sie als Multithread-Anwendung bezeichnet.
Starten Sie parallel dazu den Task Manager oder ähnliches und beobachten zur Laufzeit, wie die Threads sich im Fenster instanziieren.
Zusatzfrage: Wie ist es möglich, die Geschwindigkeit eines Systems technisch zu reduzieren?

10. Virtueller Speicher Test

Hier wollen wir das Stressen der Auslagerungsdatei und der CPU testen.
Unter Tel_2\dpatterns_code\ finden Sie das Programm patterns.exe

- Wo befindet sich die Auslagerungsdatei im OS und was bedeutet File Caching
- Welche Einstellmöglichkeiten bezüglich Swapping sind möglich
- Wo im Code befindet sich die Möglichkeit, Speicher zu allozieren (Memory Stress) und Speicher freizugeben (Free Memory) und welche DLL ist verantwortlich für die globale Speicherverwaltung.

11. Zugriff auf die API zum Darstellen von Prozessen

Im weiteren wollen wir unseren Client mit einer Prozessdarstellung erweitern. Eigentlich sollten wir die zusätzlichen Funktionen in die DLL einbauen, da nur so andere Clients die Dienste auch extern nutzen können.
Die benötigte Datei aus der wir die Funktion einbauen lässt sich aus untenstehendem Code produzieren und benötigt die Unit Thelp32.

```
lsvProcess.Items.Clear;
  FSnapshotHandle:= CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0);
```

```
   FProcessEntry32.dwSize:= Sizeof(FProcessEntry32);
   ContinueLoop:=Process32First(FSnapshotHandle,FProcessEntry32);
   while integer(ContinueLoop)<>0 do begin
        NewItem:= lsvProcess.Items.add;
        NewItem.Caption:= ExtractFileName(FProcessEntry32.szExeFile);
        NewItem.subItems.Add(FProcessEntry32.szExeFile);
        ContinueLoop:= Process32Next(FSnapshotHandle,FProcessEntry32);
   end;
 CloseHandle(FSnapshotHandle);
```

Zusätzlich benötigt man die Komponente TListView mit den Eigenschaften:
ViewStyle: vsReport und Columns: mit den Titeln "Name" und "Pfad"

12. Symmetrischer Schlüssel bauen (Kryptologie)

Benötigt wird auf unserem Client ein weiteres Label, die eigentlichen Funktion Encode können wir in die DLL auslagern (ja nach Kenntnis), der wir den String übergeben und wieder zurückholen:
Frage: wo befindet sich der Schlüssel und warum ist keine Funktion „Decode“ nötig?

```
procedure TForm1.Button1Click(Sender:TObject);
var
  s: string[255];
  c: array[0..255] of Byte absolute s;
  i: Integer;
begin
  {encode}

  s:= Label1.Caption;
  For i:= 1 to ord(s[0]) do c[i]:= 23 XOr c[i];
  Label1.Caption:= s;
end;
```

13. Manipulieren der Registry

Hier geht es um das direkte Lesen aus der Registry im OS:
Lesen Sie zusätzlich einen anderen Wert aus der Reg., z.B. die Internet_Explorer\TypedURLs

```
Function TForm1.RegRead (keyPath, myField: String): string;
begin
//Create the Object
  with (TRegistry.Create) do begin //als Instanz
    RootKey:=HKEY_LOCAL_MACHINE;
    //Check if whe can open key, if the key dosn't exist, we create it
    if OpenKey(keyPath,true) then begin
      if ValueExists(myField) then
      result:= ReadString(myField) else
      ShowMessage(myField+' does not exists under '+keyPath);
    end else
      ShowMessage('Error opening/creating key : '+keyPath);
     closeKey;
    Free;
  end; //with
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Label1.Caption := RegRead('NETWORK\logon','username');
end;
```

14. Synchrones oder asynchrones Starten einer Ressource

Schritt 1:
Erstelle eine Script Datei (*.RC) mit einem
Text Editor (z.B Notepad) und füge folgende Linie hinzu:

```
1 WAVE "MyWav.wav"

Die '1' gibt den Index der Ressource an.
WAVE' gibt an, dass es sich um eine Wave Datei handelt.
Und schliesslich gibt der dritte Eintrag den Namen der
Wave Datei an. Nun die Datei als MyWav.RC speichern.
```

Schritt 2:

```
Mit Borland's Resource Compiler, BRCC32.EXE, wird
die Datei in eine .RES Datei compiliert.
Im MSDOS Prompt, im Verzeichnis wo sich MyWav.RC befindet, nun folgendes ei
ngeben: BRCC32 MyWav.RC
Dies erzeugt eine Ressource-Datei MyWav.Res
```

Schritt 3:

```
Nun muss noch eine Compiler Directive dem Sourcecode hinzugefügt werden.
Der Eintrag sollte unmittelbar nach der Form Directive folgen, so wie hier:

{$R *.DFM}
{$R MyWAV.RES}
```

Schritt 4:

```
Füge noch folgenden Code dem Client hinzu:
procedure TForm1.Button1Click(Sender: TObject);
begin
  PlaySound(PChar(1),HInstance, snd_ASync or snd_Memory or snd_Resource);
end;

Es können natürlich auch mehrere .Wav Dateien in eine ExeDatei eingebunden
werden. Einfach eine andere Index Nummer nehmen und anstatt
PChar(1) PChar(Index) schreiben. Frage: Wie lautet der synchrone Aufruf?
Tipp: Noch die Unit MMSystem  der Uses Klausel hinzufügen!
```

15. Zugriff auf das Filesystem mit Streams

Streams sind eine schnelle Form von Dateitransfer als Datenstrom. Streaming Media ist eine Internet-Technologie, die es erlaubt, Audio-, Video- oder andere Multimediadaten im Internet oder in Intranets in Echtzeit zur Verfügung zu stellen. Neu an der Streaming-Technik ist, dass lange Wartezeiten entfallen. Die Daten können direkt abgespielt werden. Der Bewegtbildcontent steht dem User weltweit zur Verfügung.

```
function GetTextFromFile(AFile: String; var ReturnString: string):boolean;
var
  FileStream: TFileStream;
begin
  result:= false;
  if not fileExists(AFile) then exit;
  FileStream:= TFileStream.Create(AFile, fmOpenRead);
  try
    if FileStream.Size > 0 then  begin
      SetLength(ReturnString, FileStream.Size);
      FileStream.Read(ReturnString[1], FileStream.Size);
      result:= true;
    end;
  finally
    FileStream.Free;
  end; //try
end;

procedure TForm1.Button1Click(Sender: TObject);
var s: string;
begin
 if GetTextFromFile('c:\autoexec.bat',s) then begin
   ShowMessage(s);  // Memo1.text:= s;
```

```
 end;
end;
```

16. Fileattribute

Diese Übung zeigt wie sich in einem Verzeichnis die vorhandenen Files zählen lassen. Die Funktion lässt sich ohne Parameter aufrufen, Rückgabewert ist die Zahl der gefundenen Files:

```
Function TForm1.getFileCount: integer;
var
 DOSerr: integer;
 fsrch: TsearchRec;
begin
  result:= 0;
  doserr:= FindFirst('*.*',faAnyFile, fsrch);
  if (DOSerr = 0) then begin
    while (DOSerr = 0) do begin
      if (fsrch.attr and faDirectory) = 0 then inc(result);
      DOSerr:= findnext(fsrch);
    end;
   findClose(fsrch);
  end;
end;
```

17. Finden der Ordnergrösse

Implementieren Sie diese Funktion und fügen Sie Kommentare hinzu, was die einzelnen Programmschritte überhaupt machen, d.h. wie funktionieren sie intern:

```
function GetFolderDate(Folder: String) : TDateTime;
var
  Rec: TSearchRec;
  Found: integer;
  Date: TDateTime;
begin
  if Folder[length(folder)]= '\' then
    Delete(Folder, length(folder), 1);
  Result:= 0;
  Found:= FindFirst(Folder, faDirectory, Rec);
  try
    if Found = 0 then begin
      Date:= FileDateToDateTime(Rec.Time);
      result:= Date;
    end;
  finally
    sysutils.FindClose(Rec);
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  d: TDateTime;
begin
  d:= GetFolderDate('C:\WINNT');
  ShowMessage(FormatDateTime('dddd, d. mmmm yyyy, hh:mm:ss', d));
end;
```

18. Darstellen von Modulabhängigkeiten mit einem CASE-Tool

Auf dem Verzeichnis CDROM\Tools\ModelMaker:
Installieren wir kurz das Demoprogramm, um dann ein Package-Diagramm zu erstellen. Nach der Installation lassen sich die Moduldateien (Units) des Patterns.exe einlesen und dann visualisieren. Diese Visualisierung dient als Voraussetzung für den IT-Architekten und dem Systemadministrator, Updates, Wartungen und Releasewechsel besser planen zu können.

19. Starten einer KDE-Anwendung in Linux

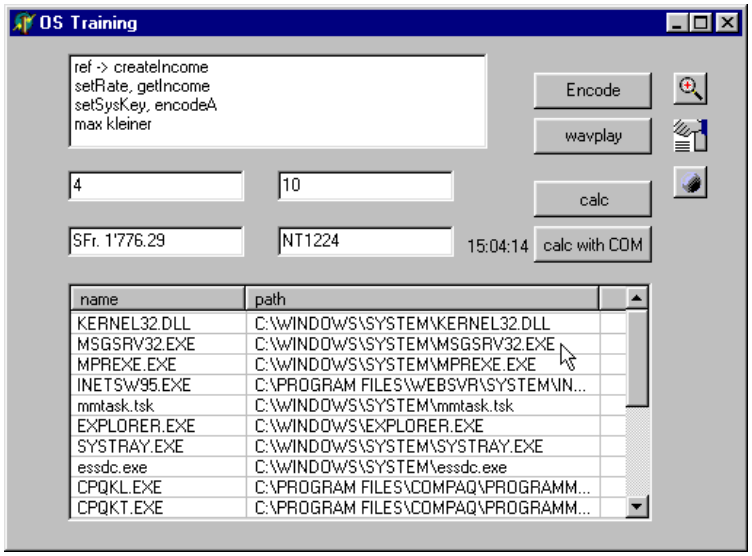
Wie starten Sie unter Linux eine Anwendung aus einem Programm ?
Welchen Unterschied gibt es zwischen den Windows-DLL's und den SharedObjects (SO) unter Linux und was heisst statisch und dynamisch an ein Programm gebunden oder gelinkt.

```
uses libc;

procedure TForm1.Button1Click(Sender: TObject);
begin
  //Execute kcalc - A calculator for KDE
  libc.system('kcalc');
end;
```

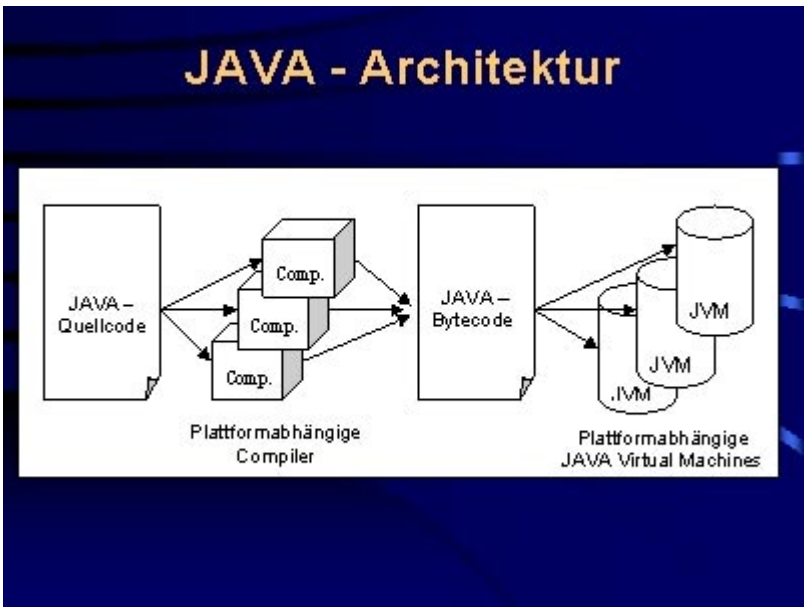
20. Installationsroutine mit InstallShield

Das Erstellen einer standardisierten Installationsroutine ist mit InstallShield mit einem Assistenten ähnlich einer Checkliste möglich. Inventarisieren Sie die benötigten Dateien und definieren ein Benutzkonzept, wie unser Client installiert werden soll.
Zum Inventarisieren benötigt der Administrator meist ein Package-Diagramm; Zum Verdeutlichen der Systemumgebung wird ein Deployment eingesetzt. Studieren Sie die Diagramme Package-Component und Deployment in einer vernetzten Umgebung und überlegen sich, wie der RedHat Packet Manager (RPM-Format) weiss, welche Abhängigkeiten zu anderen Bibliotheken oder Objekten bestehen. Hierzu benötigen Sie Linux-Kentnisse.
Aufgaben: Welche Möglichkeiten gibt es, von laufenden Programmen die Abhängigkeiten zu anderen Programmen festzustellen?
Welche Techniken gibt es, Programme via LAN automatisch installieren zu lassen?
Welche Tools gibt es, die ein fernsteuern von Programmen erlauben?



Diskutieren Sie die Vor- und Nachteile einer Virtual Machine

Ein virtueller Rechner ist im Browser oder im OS integriert. Der Java Byte-Code ist plattform-unabhängig und wird durch die plattformabhängige VM interpretiert. Das war auch bei UCSD Pascal der Fall. Früher hiess es einmal für den geneigten Entwickler: Write once run anywhere, heute heisst es eher Write once test anywhere ;):



Java, C# and ObjectPascal [iii]
or The new ideological quarrels
The news that Microsoft will not support Java any more seems to be a new chapter in the war between the monopolist to this day and the free world. This is a severe disadvantage for Java developers as the Internet Explorer 6 will not contain a Java virtual machine and as Java will not be supported in .NET.

But .NET will bring along insecurities also for Visual Basic developers, because .NET compatibility requirements will produce major changes in the next release of Visual Basic.
C++ looks like a phaseout model to me and C# may drown with .NET. I think the real winner will be Delphi. With Delphi's Kylix a true compiler is available which supports the Windows and the Linux platform.

21. Quersumme berechnen

Mit Hilfe dieser Funktion kann man die Quersummer einer Zahl errechnen. Die Quersummer erhält man, wenn man alle Ziffern addiert.

```
Function Quersumme(i : integer): integer;
var
p: pchar;
begin
 result:= 0;
 p:= Pchar(inttostr(i));
 while (p^ <> #0) do
 begin
   result:= result + strtoint(p^);
   p:= CharNext(p)
 end;
end;
```

Aufgerufen werden kann die Funktion Beispielsweise folgendermaßen:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Caption:=IntToStr(Quersumme(122));
end;
```

22. Fakultät berechnen

Das Ergebnis würde an dieser Stelle 5 sein.
Die Fakultät einer Zahl wird nach folgenden Schema ausgerechnet:
Fakultät von 10:
10! = 1*2*3*4*5*6*7*8*9*10

Diese Funktion berechnet die Fakultäten von Integerzahlen. Beachte aber, dass der Parameter nicht zu hoch gewählt wird, damit der Wertebereich des Integers nicht zu schnell überläuft.

```
Function Fakultaet(Zahl : integer): Integer;
begin
  If Zahl = 0 then
    result:= 1
  else
    result:= Zahl * Fakultaet(Zahl - 1);
end;
```

Aufgerufen werden kann die Funktion Beispielsweise so:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Caption:= IntToStr(Fakultaet(10));
end;
```

Als Ergebnis gibt die Funktion in diesem Falle ' 3628800' zurück.

23. Ist eine EXE gestartet ?

Folgende Funktion überprüft, ob eine bestimmte EXE-Datei gestartet wurde. Als Parameter muss der Funktion der Pfad+Dateiname übergeben werden. Es wird versucht eine Datei mit dem gleichen Dateinamen anzulegen.

```
function ExeFileIsRunning(ExeFile: string): boolean;
var
  H:word;
begin
  H:= CreateFile(PChar(ExeFile), GENERIC_READ,
                 0, NIL, OPEN_EXISTING, 0, 0);
  Result:= (H >= 65535);
  CloseHandle(H);
end;
```

24. Download einer HTML-Datei

Delphi 5 liegen die FastNet Komponenten bei. Die sind leider sehr fehlerhaft. In Delphi 6 und in Kylix werden deshalb die Komponenten nicht mehr dabei sein. Sie werden durch die Indy-Komponenten ausgetauscht. Downloaden kannst Du sie unter http://www.nevrona.com/indy/
Im folgenden Beispiel will ich zeigen, wie man ganz einfach eine HTML-Datei mit Hilfe der Indy-Komponenten downloaden kann.
Zuerst muss die Unit IdHTTP der USES-Klausel hinzugefügt werden.

```
function GetURLByIndy(const AURL: String): String;
var
  IdHTTP: TIdHTTP;
begin
  IdHTTP:= TIdHTTP.Create(nil);
  try
    Result:= IdHTTP.Get(AURL);
  finally
    IdHTTP.Free;
  end;
end;
```

Als Parameter muss der Funktion die URL, der zu downloadenen Datei angegeben werden.
Aufgerufen kann die Funktion Beispielsweise so:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Text:= GetURLByIndy('http://www.delphi-treff.de');
end;
```

25. IP-Adresse via Computernamen herausfinden

Die untenstehende Routine liest die IP-Adresse aus dem Computernamen aus. Der entscheidende Bestandteil ist die Funktion GetHostByName.

```
Uses
  WinSock;

FUNCTION GetIpAddressByName(const AComputerName: STRING): STRING;
VAR
  TMPResult: STRING;
  WSA: TWSAData;
  H: PHostEnt;
  P: PChar;
BEGIN
  IF WSAStartUp($101, WSA) = 0 THEN
    BEGIN
      GetMem(P,255 + 1);
      StrPCopy(P, ComputerName);
      H:= GetHostByName(P);
      FreeMem(P);
      IF H <> NIL THEN
        BEGIN
          P:= inet_ntoa(PInAddr(H^.h_addr_list^)^);
          TMPResult:= StrPas(P)
        END;
      WSACleanUp;
      IF TMPResult <> '' THEN
        Result:=TMPResult
      ELSE
          Result:= '0';
    END;
END;
```

Ein Aufruf könnte so aussehen:

```
Procedure Tform1.Button1Click(Sender: Tobject);
Begin
  Edit2.Text:= GetIpAddressByName(Edit1.Text);
End;
```

26. TStrings verknüpfen

Wenn man zwei normale Variablen hat und diese miteinander verknüpfen möchte macht man dies meistens so:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Str1: String;
  Str2: String;
begin
  Str1:= 'Test-String';
  Str2:= Str1;
end;
```

In Str2 steht dann genau das Gleiche drin, wie in Str1. Verändert man jetzt Str1, wird der Wert in Str2 nicht verändert.
Wenn man jetzt Beispielsweise zwei StringListen hat und diese auch miteinander verknüpfen möchte würde man vielleicht am Anfang zur folgenden Lösung greifen:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  StrL1: TStringList;
```

```
  StrL2: TStringList;
begin
  StrL1:= TStringList.Create;
  StrL2:= TStringList.Create;
  try
    StrL1.Add('Erster String');
    StrL1.Add('Zweiter String');
    StrL1.Add('Dritter String');
    StrL2:= StrL1;  //Hier ist die entscheidende Zeile
  finally
    StrL1.Free;
    StrL2.Free;
  end;
end;
```

Es wird aber nur die Speicheradresse übergeben und die Einträge werden nicht in die andere StringList kopiert. Dies kann man schnell nachprüfen, indem man aus der ersten Liste einen Wert löscht. In der zweiten Liste wird der Wert dann auch gelöscht. Es kann auch zu anderen Problemen kommen (in meinem Test verschwand jetzt gerade das Programmfenster).
Die Lösung für dieses Problem sieht wie folgt aus:
StrL2.Assign(StrL1);

27. Anzahl der Prozessoren ermitteln

Die untenstehende Function ermittelt die Anzahl der installierten CPUs im System:

```
function GetNumberOfProcessors: longint;
var
  SystemInfo: TSystemInfo;
begin
  GetSystemInfo(SystemInfo);
  Result:= SystemInfo.dwNumberOfProcessors;
end;
```

28. Bitmap in ein StringGrid zeichneni

Das folgende Beispiel zeigt, wie man ein Bitmap (oder eine beliebige andere Grafik) in eine Zelle eines StringGrids zeichnen kann.

Im Ereignis OnDrawItem des StringGrids sollte ungefähr folgendes stehen:

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  if ARow = 0 then
    StringGrid1.Canvas.Draw(Rect.Left,Rect.Top, Image1.Picture.Bitmap);
end;
```

In diesem Beispiel wird in jede Zelle der erste Zeile ein Bild gezeichnet und zwar das Bitmap, welches in Image1 enthalten ist. Die ersten beiden Parameter der Draw-Procedure geben die linke und die obere Position an.
In Delphi 3 muss ARow durch Row ersetzt werden.

29. Drucken eines Bildes

Die Antwort darauf ist relativ einfach.
Zuerst muss die Unit für den Drucker eingebunden werden:

```
uses ... {andere Units}, Printers;
```

Nun, wenn ein Bild gedruckt werden soll, folgende Procedure aufrufen:

```
procedure PrintBitmap(aGraphic: TGraphic, Title: string);
begin
  Printer.Title:= Title;
  Printer.BeginDoc;
  Printer.Canvas.Draw(0,0,aGraphic);
  Printer.EndDoc;
end;
```

Der erste Parameter ist das zu druckende Bild. Dies kann sowohl von Typ TBitmap, TIcon, TMetaFile, oder aber auch von anderen Nachfahren von TGraphic sein, wie TJPEGImage aus der Unit JPEG.
Der zweite Parameter gibt der Titel des Druckauftrags an, der im Druckmanager angezeigt wird.

30. Zeitmessung durchführen

Es tritt immer wieder das Problem auf, dass Du mehrere Möglichkeiten vorliegen hast, aber nicht weißt, welcher Algorithmus nun letztendlich schneller ist.

Mit Hilfe von GetTickCount kann man ganz gut eine einfache Zeitmessung durchführen. Diese Zeitmessung kann aber verfälscht werden, wenn z.B. im Hintergrund rechenintensive Anwendungen ablaufen oder Daten geswappt werden müssen und und und.
GetTickCount gibt die Zeit in ms zurück, die seit dem letzten Windowsstart vergangen sind. Der Rückgabewert ist vom Typ DWORD.

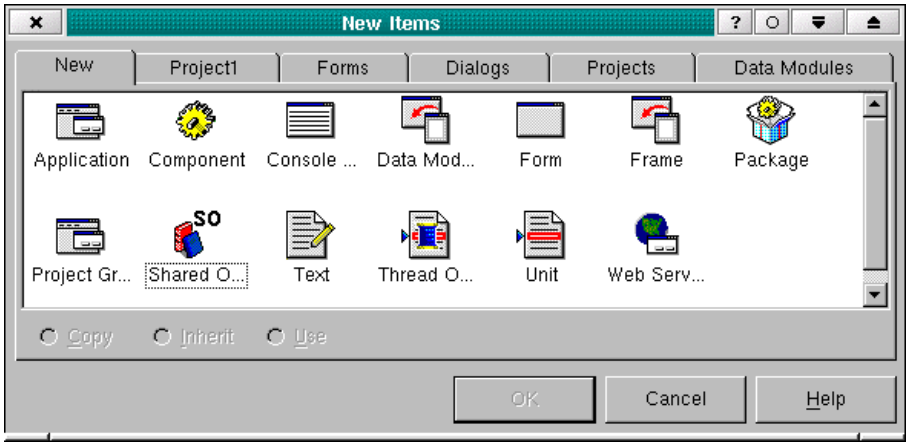
```
Procedure Tform1.Button1Click(Sender: TObject);
var
  Zeit: DWORD;
begin
  Zeit:= GetTickCount;

  //Hier kommen die Befehle rein, von denen die Zeit gemessen soll.

  Caption:= IntToStr(GetTickCount-Zeit);
end;
```

Ganz zum Schluss muss der Wert noch ausgegeben werden. Das Ergebnis wird in unserem Falle in der Titelleiste ausgegeben. Zuvor muss noch die Differenz zwischen dem jetzigen Wert und dem Startwert gebildet werden.

Have a lot of Fun with Pascal



[i] Kleiner Max et al., Patterns konkret, Software&Support 2003
ii http://www.softwareschule.ch/ostraining.zip
[iii] http://www.delphi3000.com Die Site für den Entwickler