****

Universal embedding of files in Delphi units

****

This article attempts to explain how to include files inside a Delphi unit / application as different kinds of binaries and how to manage them without the resource technology.

The app is called the Hexer works on VCL and CLX.

//Update 1, 2005-05-29: Migration to Linux in one pas file
//Update 2, 2005-06-05: Call the bytearray from a dll (export)

You can put your files into your delphi project at design time and then just compile it to get all into one or more units.
First you have to convert the file in a delphi unit by the Hexer. Then you call it from your own app which embedds the unit.
You can even store waves or pictures and play it without the need to create a file, play it from memory.

```
if not PlaySound(@UtopiaWinstarten_wav, 0, SND_MEMORY + SND_SYNC)
```

It is possible to embed any kind of file (also executables) in an executable using the Hexer. First example is a picture:

```
procedure TForm1.btnPicloadClick(Sender: TObject);
var mybitmap: TBitmap;
    bitStream: TStream;
begin
  //mybitmap.LoadFromFile('dws_logo.bmp'); for the mortals
  mybitmap:= TBitmap.Create;
  bitStream:= TMemoryStream.Create;
  try
    bitStream.Writebuffer(dws_logo_bmp, sizeof(dws_logo_bmp));
    bitStream.Position:= 0;
    mybitmap.LoadFromStream(bitStream);
    if assigned(mybitmap) then begin
      image1.Picture.assign(mybitmap);
      image1.update;
    end;
  finally
    bitStream.Free;
    mybitmap.Free;
  end;
end;
```

Second example is an executable:

```
procedure TForm1.btnApploadClick(Sender: TObject);
var mybitmap: TBitmap;
    bitStream: TMemoryStream;
begin
  mybitmap:= TBitmap.Create;
```

```
  bitStream:= TMemoryStream.Create;
  try
    bitStream.Writebuffer(viergewinnt_exe, sizeof(viergewinnt_exe));
    bitStream.Position:= 0;
    bitstream.LoadFromStream(bitstream);
    bitstream.SaveToFile('vierg.exe');
    WinExec(pchar('vierg.exe'){ + ' ' + ParamStr},SW_NORMAL);
  finally
    bitStream.Free;
    mybitmap.Free;
  end;
end;
```

How does the Hexer works?

From the view of the stack it looks like:

```
MainForm2.TForm1.btnConvertClick (97i)
  MainForm2.TForm1.ConvertToUnit (103i)
    MainForm2.WriteString (108i)
    MainForm2.EmbedFile (117i)
      MainForm2.WriteString (108i)
```

We open with createFile() the unit for write down the file in the unit. The CreateFile function creates or opens objects and returns a handle that can be used to access the object.

```
  f_hndoutput:= CreateFile(PChar(ChangeFileExt(edtUnitToCreate.Text,
                  '.pas')), GENERIC_WRITE, 0, NIL, CREATE_ALWAYS,
                                      FILE_ATTRIBUTE_NORMAL, 0);
```

Then we call EmbedFile which opens the file to be write down in the unit. The WriteFile function writes data to a file and is designed for both synchronous and asynchronous operation. The function starts writing data to the file at the position indicated by the file pointer.

```
BOOL WriteFile(
    HANDLE hFile, // handle to file to write to
    LPCVOID lpBuffer, // pointer to data to write to file
    DWORD nNumberOfBytesToWrite, // number of bytes to write
    LPDWORD lpNumberOfBytesWritten, // pointer to bytes written
    LPOVERLAPPED lpOverlapped );
```

After the write operation has been completed, the file pointer is adjusted by the number of bytes actually written.

```
    f_hndopen:= CreateFile(PChar(filename),  GENERIC_WRITE +
                  GENERIC_READ, 0, NIL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
```

Each time the file pointer is adjusted we write a block of

```
abuf: array[0..19] of byte;
Result:= ReadFile(f_hndopen, abuf, sizeof(abuf), BytesRead, NIL);
          if bytesRead > 0 then begin
```

At least our generated unit looks like

```
Unit dws_logo2;

interface

const
   dws_logo_bmp: array[0..21717] of byte = (
$42,$4D,$D6,$54,$00,$00,$00,$00,$00,$00,$76,$00,$00,$00,$28,$00,$00,$00,$F0,$00,
$00,$00,$B4,$00,$00,$00,$01,$00,$04,$00,$00,$00,$00,$00,$60,$54,$00,$00,$C4,$0E,
$00,$00,$C4,$0E,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,
..............
----------------------------------------------------------------
```

The app der Hexer

```
unit MainForm2;
{Analyzed by:           PAL - Pascal Analyzer version 2.1.9.1
Parse speed:            186 lines in 0.11 seconds (1691 lines/sec).
Main file:              D:\FRANKTECH\DELPHMAX\HEXER\MAINFORM2.PAS
Compiler:               Delphi 7.1, ModelMaker 6.2}

interface

uses
  Windows, Classes, Controls, Forms, Dialogs, ExtCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    btnConvert: TButton;
    lbFilesToConvert: TListBox;
    btnAddFiles: TButton;
    btnDeleteFiles: TButton;
    edtUnitToCreate: TEdit;
    btnBrowse: TButton;
    Bevel1: TBevel;
    lblfiles: TLabel;
    lblunit: TLabel;
    OpenDialog: TOpenDialog;
    SaveDialog: TSaveDialog;
    btnExit: TButton;
    procedure FormCreate(Sender: TObject);
    procedure btnAddFilesClick(Sender: TObject);
    procedure btnBrowseClick(Sender: TObject);
    procedure lbFilesToConvertClick(Sender: TObject);
    procedure btnDeleteFilesClick(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure btnConvertClick(Sender: TObject);
  private
    { Private declarations }
    procedure CheckButtonStatus;
    procedure ConvertToUnit;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
uses sysutils;

const
  CRLF = #13#10;
  CRLF2 = #13#10#13#10;
  STR3L = '   ';

procedure TForm1.CheckButtonStatus;
```

```
begin
  btnConvert.Enabled:= (edtUnitToCreate.Text <> '') and
                       (lbFilesToConvert.Items.Count > 0);
  btnDeleteFiles.Enabled:= lbFilesToConvert.SelCount > 0;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckButtonStatus;
end;

procedure TForm1.btnAddFilesClick(Sender: TObject);
var x : integer;
begin
  if OpenDialog.Execute then
    if OpenDialog.Files.Count > 0 then
      for x:= 0 to OpenDialog.Files.Count-1 do
        if not lbFilesToConvert.Items.IndexOf(OpenDialog.Files[x]) > -1 then
          lbFilesToConvert.Items.Add(OpenDialog.Files[x]);
    CheckButtonStatus;
end;

procedure TForm1.btnBrowseClick(Sender: TObject);
begin
  if SaveDialog.Execute then
    edtUnitToCreate.Text:= SaveDialog.FileName;
  CheckButtonStatus;
end;

procedure TForm1.lbFilesToConvertClick(Sender: TObject);
begin
  CheckButtonStatus;
end;

procedure TForm1.btnDeleteFilesClick(Sender: TObject);
begin
  lbFilesToConvert.DeleteSelected;
  CheckButtonStatus;
end;

procedure TForm1.btnExitClick(Sender: TObject);
begin
  Close;
end;

procedure TForm1.btnConvertClick(Sender: TObject);
begin
  //procedure could be moved to a logic unit
  ConvertToUnit;
end;

procedure TForm1.ConvertToUnit;
var f_hndoutput: THandle;
    cntr: integ er;
    f_error: boolean;

  function WriteString(const theString: string) : boolean;
  var bytesToWrite,
      bytesWritten: cardinal;
  begin
    bytesToWrite:= Length(theString);
    WriteFile(f_hndoutput, theString[1], bytesToWrite, bytesWritten, NIL);
    Result:= bytesToWrite = bytesWritten;
  end;

  function EmbedFile(const filename: string): boolean;
  var f_hndopen: THandle;
      abuf: array[0..19] of byte;
      bytesRead, x,
      fSizeHigh, fSizeLow: cardinal;
```

```
      fSize: int64;
      fExtension: string;
  begin
    f_hndopen:= CreateFile(PChar(filename),  GENERIC_WRITE +
                GENERIC_READ,0, NIL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
    Result:= f_hndopen <> INVALID_HANDLE_VALUE;
    if Result then begin
      fSizeLow:= GetFileSize(f_hndopen, @fSizeHigh);
      Result:= (fSizeLow <> $ffffffff) or (GetLastError = NO_ERROR);
      if Result then begin
        fSize:= int64(fSizeHigh) * $100000000 + fSizeLow;
        fExtension:= ExtractFileExt(filename);
        if Length(fExtension) > 0 then
          Delete(fExtension,1,1);
        Result:= WriteString(STR3L +
                  ExtractFileName(ChangeFileExt(filename,'_'+fExtension)) +
                  ': array[0..' + IntToStr(fSize-1) + '] of byte =
                      (' + CRLF);
        if Result then begin
          repeat
            //bytesToRead:= sizeof(buffer);
            Result:= ReadFile(f_hndopen, abuf, sizeof(abuf), bytesRead, NIL);
            if bytesRead > 0 then begin
              Result:= Result and WriteString(STR3L);
              for x:= 0 to bytesRead-1 do begin
                if x > 0 then
                   Result:= Result and WriteString(',');
                Result:= Result and WriteString('$'+IntToHex(abuf[x], 2));
              end;
              fSize:= fSize - bytesRead;
              if fSize > 0 then
                Result:= Result and WriteString(',' + CRLF);
            end;
          until (bytesRead = 0) or not Result;
          Result:= Result and WriteString(');' + CRLF2);
        end;
      end;
      CloseHandle(f_hndopen);
    end; //result
  end;
begin
  f_hndoutput:= CreateFile(PChar(ChangeFileExt(edtUnitToCreate.Text,
                   '.pas')), GENERIC_WRITE, 0, NIL,
                   CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0);
  f_error:= false;
  if f_hndoutput <> INVALID_HANDLE_VALUE then begin
    Screen.Cursor:= crHourGlass;
    if WriteString('Unit ' +
                ExtractFileName(ChangeFileExt(edtUnitToCreate.Text,
                        '')) + ';' + CRLF) then begin
      if WriteString(CRLF + 'interface' + CRLF2) then
        if WriteString('const ' + CRLF) then begin
          for cntr:= 0 to lbFilesToConvert.Items.Count-1 do
             if EmbedFile(lbFilesToConvert.Items[cntr]) then
          if WriteString(CRLF + 'implementation' + CRLF2) then
             f_error:= not WriteString('end.');
        end;
    end;
    Screen.Cursor:= crDefault;
    CloseHandle(f_hndoutput);
  end;
  if f_error then
    ShowMessage('Some Errors occured');
end;

end.
```

Conclusion:

You do have 3 possibilites to embed files:
1. article 2606 or 4217 shows the way with resources / resource workshop
2. article 2321 is based on TComponent and TStream with write and read
3. this article is more generic the advantage is transparency at design time and protection at runtime of the embedded files, you can even use binaries which you control with an inline assembler format.
The whole project Hexer and loader/player is downloadable.

Update 1
Some points to CLX:

Despite the low level stuff of the winapi no great obstacles were found during the migration.

```
  QForms, QDialogs,
  QStdCtrls, QControls, QExtCtrls, Classes;
```

a few functions had to be reselected, for ex:
FileWrite writes Count bytes to the file given by Handle from the buffer specified by Buffer.
Handle is a file handle returned by the FileOpen or FileCreate method.
or another approach to define the filesize has to be considered:

{If the file is declared as a file of byte, then the record size defaults to one byte, and FileSize returns the number of bytes in the file.}

Update2
Exporting a great const from a DLL

First you declare the type of the const and second you define a wrap function around the type:

```
library hexerdll;

type Tviergewinnt_exe = array[0..88598] of byte;

const

   viergewinnt_exe: TViergewinnt_exe = (
   $4D,$5A,$00,$01,$01,$00,$00,$00,$08,$00,$10,$00,$FF,$FF,$08,$00,$00,$01,$00,$00,
   $00,$00,$00,$00,$40,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,
   $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,
   ..............


function getArrayofByte: Tviergewinnt_exe;
   begin
     result:= viergewinnt_exe;
   end;

exports
   getArrayofByte;
```

The call of the client has the following structure:
the important thing is to declare a local var of the const in our case dllexe: Tviergewinnt_exe;

```
unit playmain;
// examples to call the embedded file in the unit or the dll
........

type Tviergewinnt_exe = array[0..88598] of byte;

function getArrayofByte: Tviergewinnt_exe; external 'hexerdll';


procedure TForm1.btnApploadClick(Sender: TObject);
var bitStream: TMemoryStream;
    dllexe: Tviergewinnt_exe;
begin
  bitStream:= TMemoryStream.Create;
  //getArrayofByte
  dllexe:= getArrayofByte;
  try
    //without a dll you call const name
   //bitStream.Writebuffer(viergewinnt_exe, sizeof(viergewinnt_exe));
    // import DLL const
    bitStream.Writebuffer(dllexe, sizeof(dllexe));
    bitStream.Position:= 0;
    bitstream.LoadFromStream(bitstream);
    bitstream.SaveToFile('viergewinnt.exe');
    case WinExec(PChar('viergewinnt.exe'), SW_SHOWDEFAULT) of
      0: ShowMessage('The system is out of memory or resources.');
      ERROR_BAD_FORMAT: ShowMessage('The .EXE file is invalid (non-Win32.EXE or error in
.EXE image).');
      ERROR_FILE_NOT_FOUND: ShowMessage('The specified file was not found.');
      ERROR_PATH_NOT_FOUND: ShowMessage('The specified path was not found.');
    end;
  finally
    bitStream.Free;
  end;
end;
```

http://www.softwareschule.ch/download/hexer2.zip

max@kleiner.com