Tips & Tricks

Identifying 32-Bit Windows Platforms

Sometimes it is necessary to code something specific to Windows NT or to Windows 95. In order to distinguish which platform you are running on, use `Win32Platform`, a new `SysUtils` variable. This has values of `Ver_Platform_Win32_Windows` for Windows 95, `Ver_Platform_Win32_NT` for Windows NT and `Ver_Platform_Win32s` for Win32s (not applicable, since Delphi 2 only generates code for NT and 95). See Listing 1. The article on calling 16-bit code from Windows 95 32-bit executables is specific to Windows 95. In the QTTHUNKU.PAS unit, home of several thunking utility routines described in the article, the initialisation section of the unit ensures that if a program using the unit runs on NT, it immediately aborts with an exception saying why.

Contributed by Brian Long

ShowModal

As a bit of a habit I (usually) like to create my modal forms on the fly. If you have many screens to simply create, `ShowModal` then free, I thought it would be nice to have the little routine shown in Listing 2. To use it simply call:

```
CreateShow(TFDistList, FDistList);
```

Contributed by Richard Smith, CompuServe 100446,327

Network Directory Exists

While working with NetWare LANs, I discovered that the routine `DirectoryExists` from the `FileCtrl` unit does not recognize the root directory of a network drive. On any subdirectories `DirectoryExists` works fine. The function in Listing 3 is a replacement for `DirectoryExists` which also works on network drives. To keep it simple, `DirExists` expects an absolute directory name as the input parameter.

Contributed by Jaroslav Blaha, CompuServe 100015,1037

Execute And Wait

I needed to start an external application (DOS, Windows, Batch, whatever) from my Delphi program and then wait until this external application terminated. The problem is to distinguish between the application started from inside the Delphi program and any other instances of the same application which might be running in the background. The function `WinExec` (from `WinProcs`) returns an undocumented `word` parameter. This parameter is the handle of the module which provided the executable code for the application. So, it cannot be used directly for the search (eg by evaluation of `GetModuleUsage`), because any other instances of this application would have the same handle. The trick is now to remember all the tasks which were running before the execution of `WinExec` and to compare them with all the tasks which are running afterwards. Any new task with the same module handle as the started application is the one we are looking for. See Listing 4.

Contributed by Jaroslav Blaha, CompuServe 100015,1037

Database Flushing

I found that if my program crashed, or maybe a laptop battery got too low, shutting the system down, my Paradox database would become corrupted. It would say *BLOB has been modified* and the program would not run anymore. Very bad news! After contacting Borland at a cost of $2/minute they told me that the following

➤ *Listing 1: How to spot a problematic platform*

```
...
type
  EThunkError = class(Exception);
initialization
  if Win32Platform <> Ver_Platform_Win32_Windows then
    raise EThunkError.Create(
      'Flat thunks only supported under Windows 95');
end.
```

➤ *Listing 2*

```
procedure TFMain.CreateShow(
  const TFClass : TFormClass; var Reference);
begin
  TForm(Reference) := TFClass.create(application);
  try
    TForm(Reference).showmodal;
  finally
    TForm(Reference).free;
  end;
end;
```

➤ *Listing 3*

```
uses SysUtils;
function DirExists(Dir : string) : boolean;
var fRec : TSearchRec;
begin
  if (Dir[ Length(Dir) ] = '\') then
    Dec(Dir[0]);                { No '\' at the end }
  case Length(Dir) of
    0, 1 : Result := false;   { Error }
    2    : Result := (Dir[2] = ':') and
                     (DiskFree(Ord(Dir[1])-64) <>
                        -1);  { Root directory }
  else
    Result := (FindFirst(Dir, faDirectory, fRec) =
      0); { Directory }
    FindClose(fRec);
  end;
end;
```

statement should be put in the `OnAfterPost` event for every `TTable`. This works very well and I have not had any corruption since! Try it and save yourself some $...

```
uses DBIProcs;
procedure TF_Main.MasterTableAfterPost(
  DataSet: TDataset);
begin
  DBISaveChanges(MasterTable.Handle);
end;
```

Contributed by mike pijl, mike_pijl@mindlink.bc.ca

DefaultDrawDataCell

Sometimes you want to highlight certain rows of a `DBGrid` or something else. At first I did it the hard way by using the `OnDrawDataCell` and drawing it myself after I had set the colours (`Default` drawing property to `Off`). Then I found I can call the `DefaultDrawDataCell` method (golden information!). See Listing 5.

Contributed by Mike Pijl, mike_pijl@mindlink.bc.ca

➤ *Listing 4*

```
uses
  WinTypes, WinProcs, ToolHelp;
procedure ExecuteAndWait(Command : string);
var
  ModuleID  : THandle;
  TaskEntry : TTaskEntry;
  TaskCount : integer;
  TaskList  : array [1..100] of THandle;
  i         : integer;
begin
  { Has to be initalized }
  TaskEntry.dwSize := SizeOf(TTaskEntry);
  TaskCount := 0;
  if TaskFirst(@TaskEntry) then
    { Save list of active tasks }
    repeat
      Inc(TaskCount);
      TaskList[TaskCount] := TaskEntry.hTask;
    until (not TaskNext(@TaskEntry));
  Command := Command + #0;
  { Execute }
  ModuleID := WinExec(@Command[1], SW_SHOWNOACTIVATE);
  if (ModuleID <= hInstance_Error) then
    { Error: add code to report it here }
  else if TaskFirst(@TaskEntry) then
    { Search for new task }
    repeat
      if (TaskEntry.hModule = ModuleID) then
        { Task uses same module }
        for i := 1 to TaskCount do
          { Search through list }
          if (TaskList[i] <> TaskEntry.hTask)
            then begin
            { Found }
            repeat
              { Wait for termination }
              Application.ProcessMessages;
            until (not IsTask(TaskEntry.hTask)) or
              Application.Terminated;
            exit;
          end;
    until (not TaskNext(@TaskEntry));
 end;
```

➤ *Listing 5*

```
procedure TF_Main.DetailGridDrawDataCell(Sender: TObject;
  const Rect: TRect; Field: TField; State: TGridDrawState);
begin
  If DetailTable.FieldByName('Row Index').AsInteger In
    [1,7,8,14] then
    with (Sender as TDBGrid).Canvas do begin
      If gdSelected in State Then
        Brush.Color:=clGreen  {further hilite cell}
      Else
        Brush.Color:=clAqua;  {highlight for weekends}
      End;
  DetailGrid.DefaultDrawDataCell(Rect,Field,State);
end;
```