Tips & Tricks

Lost Pointers

Now and then my programs generate GPFs or other exceptions (well, no-one's perfect!). Perhaps an object has not been created, or a pointer has been destroyed somewhere. Typically, Delphi then starts its built-in debugger and shows the line which generates the exception. Unfortunately, the debugger does not let me look at the variables to see who is the culprit, since the current context on the stack is lost. A simple solution is to patch a `TRY..EXCEPT..END` around the line where the failure occurs. I place a breakpoint in the exception and when the debugger reaches this breakpoint I can examine my variables to find out why the GPF occurred. Listing 1 shows a simple example.

```
function Average(i_Int1 : integer; i_Int2 : integer) :
  integer;
begin
  Result := (i_Int1+i_Int2) div 2;
end;

procedure TForm1.Button1Click(Sender: TObject);
{ Without exception }
var
  int1 : integer;
  int2 : integer;
  ptr1 : ^integer;
  ptr2 : ^integer;
begin
  int1 := 10;
  int2 := 20;
  ptr1 := @int1;
  ptr2 := @int2;
  ptr2 := nil;        { Pointer is destroyed }
  int1 := Average(ptr1^, ptr2^);  { It GPFs! }
  ShowMessage ('Average= ' + IntToStr(int1));
end;

procedure TForm1.Button2Click(Sender: TObject);
{ With exception }
var
  int1 : integer;
  int2 : integer;
  ptr1 : ^integer;
  ptr2 : ^integer;
begin
  int1 := 10;
  int2 := 20;
  ptr1 := @int1;
  ptr2 := @int2;
  ptr2 := nil;                  { Pointer is destroyed }
  try
    int1 := Average(ptr1^, ptr2^);         { It GPFs! }
    ShowMessage ('Average= ' + IntToStr(int1));
  except
    ShowMessage('Exception!'); { Put breakpoint here! }
  end;
end;
```

➤ *Listing 1*

Contributed by Goeran Forsstroem from Vaesteras, Sweden (email: goran.forsstrom@seisy.mail.abb.com)

I/O Error Handling Update

If you've been following my series on File Handling you will want to make a special note of this. Delphi 2 adds certain porting problems for I/O error handling. Although Win32 uses the same error codes as DOS, some errors are reported more precisely and so different errors get generated. As an example, a call to `ChDir('A:\')` where there is no floppy disk in drive A: will give an I/O error of 3 (`path not found`) in Delphi 1, but error 21 (`ERROR_NOT_READY, device not ready`) in Delphi 2. For a complete list of Win32 error codes load up the Delphi 2 help index and look for "error codes" and then choose "Error Codes (Win32 Programmer's Reference)" but be prepared for some missing text (many occurrences of "ERROR" are listed as "ROR").

Contributed by Brian Long (email: CompuServe 76004,3437)

Hints For Modeless Forms In DLLs

If you have a form in a DLL, then hints for the controls on the form will only show if you do a `ShowModal`, not a regular `Show` of the form. Most people believe that this might be due to the fact that a DLL has no application object (which is not true), or that a DLL cannot have a timer (also not true). The fact is, the application object of a DLL is not 'activated' like the application object of a normal application, ie it doesn't have a message loop. A message loop is important, because that will call the `OnIdle` event, which will in the end check if there is a hint to be shown. So, if we want our modeless dialogs and forms in DLLs to show hints as well, it seems we have to call the `Idle` method. Unfortunately, this method is `private`. However, a bit of digging in the VCL shows that if you put a timer somewhere on the form and make it call `HandleMessage` every time it ticks (every 1/100th second for example), then you get the hints. `HandleMessage` will call the `private` method `Idle`, which in turn will make sure the hints are shown.

Contributed by Bob Swart (email: CompuServe 100434,2072)

Is Delphi Running?

If you want to write a program or component which will only work when Delphi is running, you will need to find out if Delphi is active. My way is to add `uses ExptIntf;` to the `implementation` section of your main unit and try to call some APIs from `ToolServices`. Another way is to see if you can find the Delphi Windows themselves: this is easier, but less error-proof; just include the DelphRun unit in Listing 2 in your `uses` clause.

Contributed by Bob Swart (email: CompuServe 100434,2072)

A Delphi Scratchpad

If you want a Delphi scratchpad, you can make a unit called `begin` and then any time you want to get there, just put the cursor on the word `begin` in your source code, press `Ctrl-Enter` and voila – just the quickest way to get to a temporary notes file!

Contributed by Michael Ax of Ax-Systems

TDirectoryOutline Bug

Using the `TDirectoryOutline` component in conjunction with a `TFileListBox`, I noticed that when the root directory is selected in the `DirectoryOutline` the `FileList` is not updated correctly, but displays the previously selected directory's file list. After a little investigation I found the reason for this. The last line in the `Click` method of `TDirectoryOutline` is:

```
Directory := Items[SelectedItem].FullPath;
```

However, for the root directory, `FullPath` returns `c:` and not `c:\` and so the last directory is not changed (just as when you type `c:` from the DOS prompt). This can be easily fixed by replacing the `Click` method with the code in Listing 3.

Contributed by Roberto De Marini from Italy (email: rdemari@mbox.vol.it)

Indexing Of TParam Objects In A TQuery.Params Property

Every instance of a parameter in a parameterized SQL statement produces a separate entry in the `Params` array. Being unaware of this and using

```
Query.Params[n] AsXxxxx := ParamValue;
```

(where `AsXxxxx` is the required `TParam` method for the `TParam.value` field type) to assign values to parameters can result in several hours of hacking of the "why doesn't this query produce the right records!" variety.

For example, the SQL statement

```
select * from FOCI where
  (FOCI.'Sex'  = :ClientSex or FOCI.'Sex' = 2)
  and FOCI.'Min Age' <= :ClientAge
  and (FOCI.'Max Age' >=
    :ClientAge or FOCI.'Max Age' = 0)
  and (FOCI 'Ethnic Grp' =
    :ClientEthnic or FOCI.'Ethnic Grp' = 0)
```

produces the `Params` values of

```
Params[0].Name = 'ClientSex'
Params[1].Name = 'ClientAge'
Params[2].Name = 'ClientAge'
{not ClientEthnic as muggins here at first thought!}
Params[3].Name = 'ClientEthnic'
```

```
unit DelphRun;
interface
implementation
uses WinTypes, WinProcs;
initialization
  if (FindWindow('TApplication', 'Delphi') = 0) OR
    (FindWindow('TPropertyInspector', nil) = 0) OR
    (FindWindow('TAppBuilder', nil) = 0) then begin
    MessageBox(GetFocus,
      'Delphi 1.0x is not running!',
      'Dr.Bob says...', MB_ICONHAND OR MB_OK);
    Halt
  end
end.
```

➤ *Listing 2*

```
procedure TDirectoryOutline.Click;
var
  TempDir: string;
begin
  inherited Click;
  TempDir := Items[SelectedItem].FullPath;
  if Length(TempDir) < 3 then
    TempDir := TempDir +'\'; {Adds required backslash}
  Directory := TempDir;
end;
```

➤ *Listing 3*

```
procedure TForm1.ButtonShowParamsClick(
  Sender: TObject);
var
  i: integer;
begin
  Listbox.Items.Clear;
  for i:= 0 to (Query.ParamCount-1) do
   Listbox.Items.Add('['+ IntToStr(i)+'] '+
     Query.Params[i].Name;
end;
```

➤ *Listing 4*

Fortunately,

```
Query. ParamByName('ClientAge').AsXxxxx :=
  ParamValue;
```

sets all instances of `ClientAge` to the correct value. To check `Params` indexes and names at run time use the code in Listing 4.

Contributed by Design Consultant Martin Humby of Portchester, Hampshire.

Send In Those Tips Please!

If you have some useful tips accumulated from your long hours of development (all highly enjoyable of course!) why not share them with your fellow Delphi developers? Just drop an email to the Editor, Chris Frizelle, on 70630.717@compuserve.com or send us a disk, letter or fax. Who knows, you could even become famous!