How to call CopyFileEx and let the callback update a progress bar

> ***QUESTION***
>
> Does anyone have an example of using *CopyFileEx* with a *CopyProgressRoutine*? I have created a function that takes the same parameters as the *CopyProgressRoutine*, but when I pass it using @ or *Addr()* I get a "Variable Required" error message.

Let's assume you call *CopyFileEx* and want the callback to update a progress bar. The callback cannot be an object's method but you can use the *lpData* parameter of *CopyFileEx* to pass any kind of data to the callback, e.g. a form reference. So, if you want to serve a progress form in the callback here's a little project that does the job.

Create a new Delphi project with two forms. Name the main form *Form1* and save it as `Unit1.pas`. Name the second form *ProgressForm* and save it as `Unit2.pas`. Ensure that *ProgressForm* is not auto-created by the project. Set *ProgressForm*'s *FormStyle* property to *fsStayOnTop*.

On the main form drop a *TButton* named *CopyFileButton* and a *TOpenDialog* component named *OpenDialog*. Create an *OnClick* event handler for the button with the following code.

```
procedure TForm1.CopyFileButtonClick(Sender: TObject);
begin
  if OpenDialog.Execute then
    DoFileCopy(OpenDialog.FileName, OpenDialog.FileName + '.bak')
end;
```

Somewhere above *TForm1.CopyFileButtonClick* in the implementation section of *Unit1* add the following code:

```
function CopyCallback(TotalFileSize, TotalBytesTransferred, StreamSize,
  StreamBytesTransferred: Int64; dwStreamNumber, dwCallbackReason: DWORD;
  hSourceFile, hDestinationFile: THandle; progressform: TProgressForm ):
  DWORD; stdcall;
var
  newpos: Integer;
const
  PROCESS_CONTINUE = 0;
begin
  Result := PROCESS_CONTINUE;
  if dwCallbackReason = CALLBACK_CHUNK_FINISHED then
  begin
    newpos := Round(TotalBytesTransferred / TotalFileSize * 100);
    with progressform.Progressbar do
      if newpos <> Position then
        Position := newpos;
    Application.ProcessMessages;
  end;
end;

function DoFilecopy(const source, target: string): Boolean;
var
  progressform: TProgressForm;
begin
  progressform := TProgressform.Create(Form1);
  try
    progressform.Show;
    Application.ProcessMessages;
    Result := CopyFileEx(
      PChar(source), PChar(target), @CopyCallback,
      Pointer(progressform), @progressform.FCancel, 0
    );
 finally
   progressform.Hide;
   progressform.free;
```

```
  end;
end;
```

DoFileCopy set up the file copy operation and displays the progress dialog box (which we'll come to in a moment) and *CopyCallback* is the *CopyProgressRoutine*. This callback calculates the progress of the file copy and updates the progress bar on the progress form.

New for the progress form. Drop a *TProgessBar* named *ProgressBar* and a *TButton* named *AbortButton* on *ProgressForm*. Add a public Boolean field named *fCancel* to *TProgressForm* and add an *OnClick* event handler to *AbortButton* as follows:

```
procedure TProgressForm.AbortButtonClick(Sender: TObject);
begin
  fCancel := True;
end;
```

Add a reference to *Unit2* in the uses clause of *Unit1*.

You can now compile and run the program. Click the button in the main form to display the open file dialog box. Select a file to copy – something largish, say 60MB should enable the progress dialog to display for long enough to see what's happening. OK the dialog box and the file copy begins and the progress dialog displays, updating the progress bar as the file copy proceeds. Try clicking the progress dialog's abort button to cancel the copy.

Tip revised and made to work correctly by DelphiDabbler.

Original resource:	The Delphi Pool
Author:	Peter Below
Added:	2010-06-02
Last updated:	2011-01-20