Porting KillerApp

by Andrew McLellan

Now that Delphi 5 is imminent, the time for *Porting The Killer Application* is once again upon us. Having been through this process with every version of Pascal and Delphi for the last fifteen years, I think I know how to do it now. And besides, last time I made notes!

There is a temptation to install the new shiny version of Delphi, open KillerApp in the new shiny version and hit F9, but such is the road to disaster. This requires a little planning.

First, back up your development folder. All of it, not just the files and extensions that look interesting.

Now install Delphi 5. Continue resisting the urge to open KillerApp, open the help file and look for *Compatibility Issues* or *What's New* in the contents. Reading this will warn you of anything that will definitely be broken. Sobbing quietly, close Delphi 5 and re-open Delphi *x*.

Next identify all third party code in use by your application: all bought in, shareware and freeware libraries and components. If there are any for which you don't have the source, then the whole porting exercise is a non-starter. Although Borland claimed with Delphi 1 that the DCU version incompatibility problem that has plagued us for years would go away, it hasn't. I personally don't find this to be a bad thing, the annual housekeeping that this forces on us is good practice and, in any case, I would not use anything for which I didn't have the source.

After this, create a new application, PortTest, with your existing copy of Delphi and include *every* unit from all the libraries not produced in-house. On the form, drop one of every component used by KillerApp. Build this application (use Build All).

Now make a complete copy of your entire development folder. Then fire up Explorer and delete every DCU in your *development* folder. (If you're paranoid, rebuild KillerApp at this point, just to be sure, then delete the DCUs again). Delphi has an annoying habit of finding source code in the wrong folder, even when told explicitly where to find a unit.

Ladies and gentlemen, you may now start Delphi 5. Open PortTest in the new folder and do File | Build All. It won't. And even if it does, the list of hints and warnings will have grown. Decision time: if you decide that the benefits of using TWhizzyEdit in place of the standard TEdit don't justify the extra work required to port it, then make the change in Delphi *x* and not Delphi 5: you will be working in an environment you know and with code you understand.

You can just open KillerApp in Delphi *x*, delete all instances of TWhizzyEdit and replace them with TEdits, but it's easier to do this by editing the DFMs directly: open each form with a TWhizzyEdit, press Alt-F12, search for TWhizzyEdit and replace it with TEdit. This leaves all the working properties and event handlers in place, and recompiling shows you all the code that has been broken by the change. When KillerApp finally recompiles in Delphi *x*, manually sift through each form looking for any methods, procedures or functions *without* the dots to the left indicating that the code hasn't been compiled into your application. Now would be a good time to test it is all still working.

Now make another complete copy of your entire development folder. Fire up Explorer and delete every DCU in your *development* folder: just to make sure Delphi doesn't find the wrong file.

At this point, you should be reasonably sure that anything you didn't write is going to *port* even if it doesn't *work*. Now for your own code. Open up KillerApp in the new folder and create a new unit in the project:

```
Unit PortIt;
Interface
Const
  TODO = FALSE;
Implementation
End.
```

Build KillerApp. If anything doesn't compile, either correct the code immediately, commenting the change with // Delphi5, so that you can find it again, or comment out the code, replacing it with Assert(TODO) and including the unit PortIt.

Fire up Explorer and check that no DCUs have appeared in your development folder. If they have, then you are compiling a mixture of source code in your copy folder and your development folder. Use the Project Manager to explicitly include units from the copy folder: if Things.DCU has appeared in the development folder, then any unit that uses Things.Pas needs to be added.

Once again build KillerApp. If anything doesn't compile, either correct the code immediately, commenting the change with // Delphi5, so that you can find it again, or comment out the code, replacing it with Assert(TODO) and including the unit PortIt.

KillerApp now compiles, but isn't going to work until you've replaced all the commented-out code. Searching all the files in the project for PortIt will show you which units have problems. Commenting out the line TODO = FALSE in PortIt and rebuilding will take you to each line which needs attention. Comment every change with // Delphi 5. Only when you can remove PortIt from your application altogether do you have any chance that KillerApp is going to run.

Finally, test it rigorously!

I don't yet know what's new in Delphi 5, but I am sure it will break some existing code. And I look forward to finding out what.

Andrew McLellan is Principal Developer at Epoch Software and the author of *Rapidocs*. You can contact him at andrew@cix.co.uk