Kylix: A Technical Overview

by Dave Jewell

After a long wait, Kylix is actually here. This overview is a follow on from the 'first-look' article which I wrote a couple of months back. As always, I have focused on the issues which interest me, and I'd suggest that you look elsewhere if you want the low-down on database goodies and the like! I'll leave that sort of thing in the capable hands of Brian Long *et al*. But if you want to know whether Kylix 1.0 is worth buying and if it represents a practical and usable development system for Linux, then read on.

What's In The Box?

Kylix 1.0, in common with most other Borland products released in recent years, comes in that hazardous 'bottomless' carton: keep a firm grip on it or the contents will escape at the first opportunity, making a large dent in the floor or (if you're unlucky) your foot. Damage to the metatarsals is rendered even more likely by the presence of the weighty *Developer's Guide* which covers all aspects of Kylix programming, a smaller *Language Guide* and a *Quick Start* booklet.

Once I'd removed the aforementioned items from the box, I finally unearthed three CDs in clamshell cases hiding at the bottom of the box. The first CD contains Kylix itself, and the second is entitled *Companion Tools*. The former includes the install scripts and the kernel patches which I mentioned a couple of months ago in my 'first-look' preview. In case you don't remember why this stuff is necessary, suffice to say that Kylix significantly pushes the envelope of the Linux operating system itself, and it won't run on old Linux distributions. One of the reasons for this, according to the online documentation, is that there are a few bugs in the standard `glibc` runtime library, and in the Linux loader which is responsible for reading executables into memory. For the vast majority of Linux applications, these bugs aren't of any significance, but because of the dynamic nature of the Kylix IDE (and, specifically, the ability to load and unload design-time packages at will) it's crucial to fix these problems before trying to run the development system.

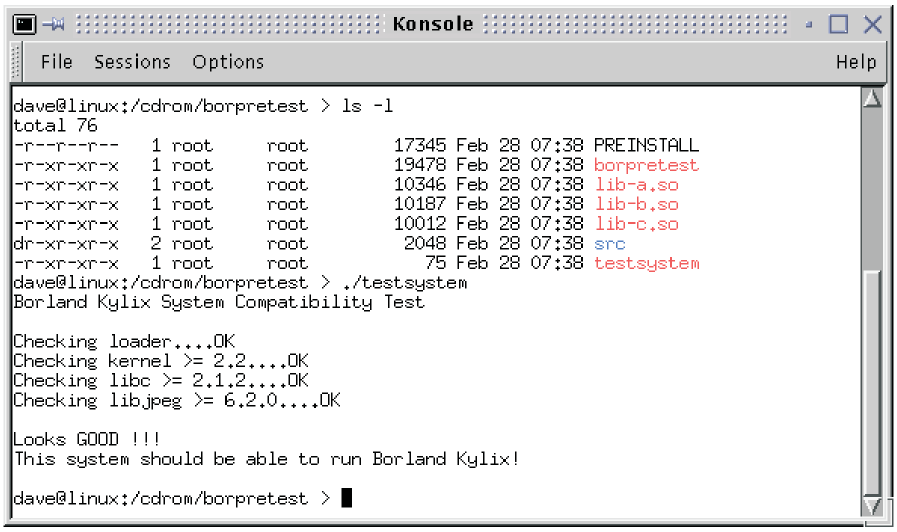
Kylix requires less than 200Mb of hard disk space, so you're not likely to need a new drive! It comes with a set of kernel patches for Red Hat 6.2 and 7.0, Mandrake 7.2 and SuSE 7.0. If you have one of these distributions, then it should be easy to get Kylix up and running; if you have an older distribution, then now's the time to think about upgrading. Just to make things easier still, you'll find an evaluation version of SuSE 7.0 bundled with Kylix: this is the third CD of the bundle. This particular version appears to already have been patched for Kylix compatibility, because when I tried running the little borpretest program (prior to installing Kylix), it immediately told me that the operating system had passed the various compatibility tests.

Another important issue is to buy some more RAM! Don't be fooled into swallowing the old adage that anything Windows does can be done under Linux using one tenth the amount of RAM, it's not so! At least, not where Kylix is concerned: it's a complex, RAD-based development tool which needs plenty of RAM to stretch out in. Borland recommend 128Mb, but I'd suggest 256Mb. Also, bear in mind that you should really double this again if you run Linux under Windows NT or Windows 2000 using VMware; in other words, if you've got a 256Mb machine, then you should give half of it to the guest operating system. Don't accept the default VMware setting which allocates a measly 48Mb to Linux, you'll find this an exercise in futility once you try and run Kylix.

Third Party Support

Delphi and C++Builder programmers are used to having the Component Palette full to bursting with assorted drop-in, reusable components. By contrast, Kylix offers a relatively spartan selection of standard controls, wrapper classes for the Qt dialogs, dbExpress and other assorted database components, and not a

➤ *Figure 1: Kylix includes a special test procedure located in the BORPRETEST directory of the main CD. This will tell you whether or not it is necessary to apply any patches to the operating system.*

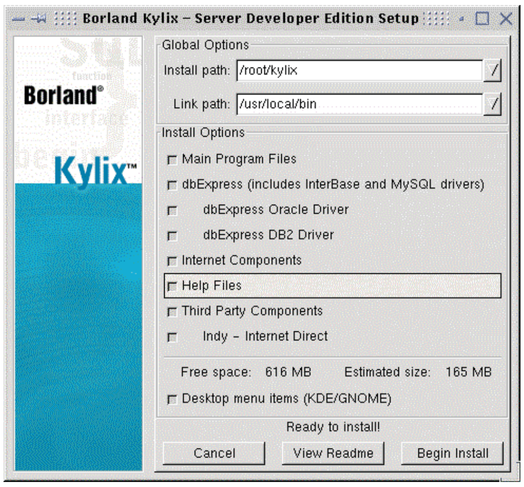


```
dave@linux:/cdrom/borpretest > ls -l
total 76
-r--r--r--   1 root     root        17345 Feb 28 07:38 PREINSTALL
-r-xr-xr-x   1 root     root        19478 Feb 28 07:38 borpretest
-r-xr-xr-x   1 root     root        10346 Feb 28 07:38 lib-a.so
-r-xr-xr-x   1 root     root        10187 Feb 28 07:38 lib-b.so
-r-xr-xr-x   1 root     root        10012 Feb 28 07:38 lib-c.so
dr-xr-xr-x   2 root     root         2048 Feb 28 07:38 src
-r-xr-xr-x   1 root     root           75 Feb 28 07:38 testsystem
dave@linux:/cdrom/borpretest > ./testsystem
Borland Kylix System Compatibility Test

Checking loader....OK
Checking kernel >= 2.2....OK
Checking libc >= 2.1.2....OK
Checking libjpeg >= 6.2.0....OK

Looks GOOD !!!
This system should be able to run Borland Kylix!

dave@linux:/cdrom/borpretest >
```


Borland Kylix – Server Developer Edition Setup

Global Options

Install path: /root/kylix

Link path: /usr/local/bin

Install Options

- Main Program Files
- dbExpress (includes InterBase and MySQL drivers)
  - dbExpress Oracle Driver
  - dbExpress DB2 Driver
- Internet Components
- Help Files
- Third Party Components
  - Indy – Internet Direct

Free space: 616 MB   Estimated size: 165 MB

- Desktop menu items (KDE/GNOME)

Ready to install!

➤ *Figure 2: The Kylix install program. (Sorry about the measles, you don't see this on the real thing!) The only real deficiency is that the program allows Kylix to be installed into the /root/kylix directory without any warning, see text.*

great deal more besides. This being the case, it's a shame that relatively little third-party support is provided 'out of the box' with Kylix. In fairness, this isn't too surprising: it's relatively early days and whilst many developers are known to be working on a Kylix port of their code, relatively few have delivered thus far. This is clear when you work your way through the offerings on the *Companion Tools* CD.

If you want to stay on top of what's available, point your newsgroup reader at newsgroups.borland.com and check out the various Kylix-specific newsgroups you'll find there. Third-party announcements can be found in the group borland.public.kylix.thirdpartytools where you'll find other interesting material such as a gentle sparring match between Brain Patchwork DX and the creators of Indy. May the best internet toolkit win!

Speaking of Indy, it's undoubtedly the star of the collection in terms of the third-party tools that Borland actually supply. Indy is a set of internet components which originally went by the very forgettable name of WinShoes. The Indy components have now been ported across to Kylix, and very nice they are too. I was a bit miffed that those gorgeous component icons have been replaced in the final release by a set of much less decorative graphics; I suspect this may be for the benefit of those who run their Linux systems with only 256-colours (or worse). Indy is great provided that you're comfortable with the 'blocking' paradigm that they use: all internet calls are blocking, which means that if you don't want to block the user interface of your (for example) newsreader application, then you'll need to move the blocking calls into another thread to maintain foreground responsiveness. At least, that's the theory, but see my comments on performance issues later in this article.

A number of third-party component vendors have announced support for Kylix, and many of these products will hopefully be available by the time you read this review. The well-known TurboPower Software Company (www.turbopower.com), for example, have announced that they plan to ship four new products for Kylix and have undertaken that everything will be available within 45 days of Kylix being released. The new goodies are: Async Professional CLX, SysTools for Linux, LockBox 2 and XMLPartner Standard Edition. LockBox 2, an encryption library, is particularly interesting to UK-based developers because until recently it wasn't available in this country due to brain-dead US export restrictions on supposedly 'sensitive' technology. As I write, hordes of Chinese scientists are allegedly crawling all over an American spy plane, so I think we can safely say that this particular horse has well and truly bolted! LockBox 2 must be good because, after all, it's exactly the same technology which TurboPower use themselves to protect their own product CDs.

The well-respected Developer Express (www.devexpress.com) are somewhat less forthcoming about Kylix thus far. Bearing in mind that these guys have already created an impressive set of drop-dead gorgeous components for Delphi and C++Builder, ported those components to ATL, making them true ActiveX controls, and are *now* working on creating C# versions of everything for .NET, it's not surprising that they have their hands full right now! Even so, Developer Express isn't a company that rests on its laurels, and in their own newsgroups (the server is at news.devexpress.com), they have recently been canvassing feedback from customers on how many of them have purchased Kylix and are actively working with it.

If the general consensus in the Developer EXpress newsgroups is representative (and of course, Borland might well argue that it's not) then most Delphi developers are either disinterested in Linux & Kylix, find the price of Kylix too high and are waiting for the Open Edition before dipping a toe into the water, or don't want to purchase Kylix until the Developer Express tools have been ported to the new development system. As the resident company representative pointed out, this represents something of a Catch-22 situation, since Developer Express are waiting for their customers to make the first move, whereas the customers are waiting for Developer Express... Despite all this, there are tantalising signs that Developer Express have already got at least some of their components working under Kylix, albeit unannounced. Try www.devexpress.com/kylix to see what I mean.

One developer who has really pulled his finger out and has something working *now* is the indefatigable David Berneda, creator of the awesome TeeChart and TeeTree components. You can track down David at www.steema.com where registered customers can download a beta of TeeChart Pro 5.0 for Kylix. Beta it may be, but it also looks very impressive, as you'll see from Figure 3. TeeTree for Kylix is apparently also in the pipeline...Yummie!

Other developers are proving even more adventurous in terms of what they've managed to achieve with Kylix, even at this early stage. I was a bit gob-smacked to discover w2kwm (see Figure 5), a replacement window manager which can be used instead of KDE, GNOME, or whatever. This project is at an early stage, and I suspect that the author is going to have to make some changes for legal reasons (at the moment it looks far too much like another operating system we know and love!) but, needless to say, I'm very impressed. Catch up with w2kwm at www.vclcrawler.com/w2kwm/ where you'll find the complete source code and binaries available for free download.

Other goodies I've found on my travels include Kamiak Tkam-NetworkAdapters, an interesting component which interrogates your network card and returns assorted metrics as a dataset. This can be found at www.kamiak.com/nwi.html. Finally, check out ProKylix, the Kylix application profiler which can be found at www.prodelphi.de. This will cost you 62.50 euros (no, I don't know what that is in real money). We'll try to bring you reviews of cool new Kylix stuff as soon as we can!

IDE Performance Issues: Is It VMware?

When evaluating a complex product like Kylix, there are a variety of different performance issues to be addressed. You need to consider the runtime performance of the native Linux executables that are created by the development system, and of course you also need to look at the behaviour of the IDE itself.

I have to be blunt with you and say that, running under VMware, the performance of the Kylix IDE was less than stellar. Initially, I installed VMware onto my main development system, a 500MHz Pentium III equipped with 256Mb of RAM. Yeah, OK, I appreciate that this isn't a cutting edge specification these days, but I'd argue that it's a pretty reasonable spec for a Linux machine (I'm moving to North Wales soon, and plan to get something a bit more leading-edge with the money that's left over!). Using VMware under Windows 2000, I installed the supplied version of SuSE 7.0 into a 2Gb virtual disk and then installed Kylix itself, configuring the virtual machine for 128Mb of RAM.

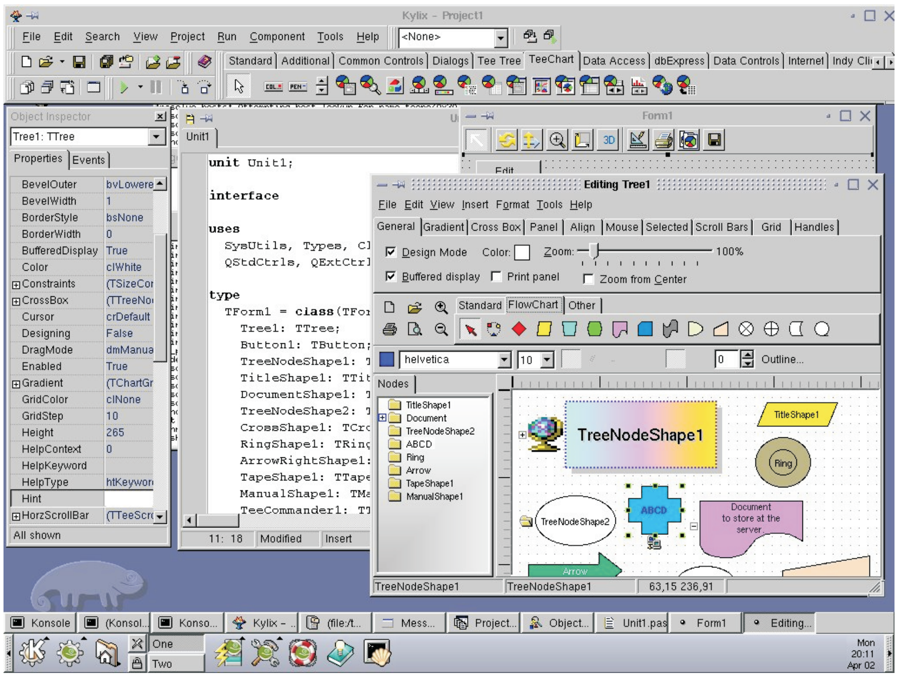
The result, I regret to say, was less than impressive. The IDE behaved like the proverbial one-legged dog in treacle. Pulling down a menu required a considerable degree of patience, and often a menu didn't appear until you'd moved along the menu bar and started trying to pull down another one! The tooltip hint windows that appear over component palette items were often incomplete, with the actual hint window having the correct width, but with up to half of the component name being conspicuous by its absence.

Worse was to come when I started trying to type code into the text editor. I don't consider myself to be enormously fast at typing, but I'm not limited to one finger either! Unfortunately, the performance of the text editor was ideally suited to a one-finger typist; whatever I typed in appeared at the rate of about one character per second, with the IDE gently ambling along behind me in a vain effort to keep up. In a word, the thing was unusable. Clearly, something was very wrong, but what?

It would obviously be unfair to simply draw a line under the Kylix IDE at that point. Frankly, I could not believe that the thing was really as bad as the performance I was seeing! My initial suspicion was that the performance problems were caused by some sort of nasty interaction with VMware, so I resolved to reinstall Kylix onto a 'native' Linux system. I have a second machine, an ageing 300MHz Pentium II, but it does have the benefit of possessing 128Mb of physical memory, so in a sense I'd be testing like with like because the aforementioned virtual machine was running with 128Mb as well.

In an effort to keep as many other factors constant as possible, I used the indispensable Partition Magic to create a 2Gb Linux partition on the Pentium II machine, installed the same version of SuSE Linux 7.0 and made sure that other configuration details remained the same. For example, I took care to install the KDE 1.x desktop, just like I'd done previously. KDE 2.x, while very pretty, is also notoriously buggy, and the last thing we

➤ *Figure 3: Here's a screenshot of a beta version of the forthcoming TeeTree for Kylix. Is this a knock-out control or what? Check www.steema.com for more details.*



Object Inspector — Tree1: TTree

BevelOuter	bvLowere
BevelWidth	1
BorderStyle	bsNone
BorderWidth	0
BufferedDisplay	True
Color	clWhite
Constraints	(TSizeCor
CrossBox	(TTreeNo
Cursor	crDefault
Designing	False
DragMode	dmManua
Enabled	True
Gradient	(TChartGr
GridColor	clNone
GridStep	10
Height	265
HelpContext	0
HelpKeyword	
HelpType	htKeywor
Hint	
HorzScrollBar	(TTeeScr

```
unit Unit1;

interface

uses
  SysUtils, Types, Cl
  QStdCtrls, QExtCtrl

type
  TForm1 = class(TFor
    Tree1: TTree;
    Button1: TButton;
    TreeNodeShape1: T
    TitleShape1: TTit
    DocumentShape1: T
    TreeNodeShape2: T
    CrossShape1: TCrc
    RingShape1: TRing
    ArrowRightShape1:
    TapeShape1: TTape
    ManualShape1: TMa
    TeeCommander1: TT
```

Kylix - Project1
File Edit Search View Project Run Component Tools Help
Standard Additional Common Controls Dialogs Tee Tree TeeChart Data Access dbExpress Data Controls Internet Indy Cli
Properties Events
All shown
11: 18 Modified Insert
Form1
Editing Tree1
File Edit View Insert Format Tools Help
General Gradient Cross Box Panel Align Mouse Selected Scroll Bars Grid Handles
Design Mode Color: Zoom: 100%
Buffered display Print panel Zoom from Center
Standard FlowChart Other
helvetica 10 0 Outline...
Nodes
TitleShape1
Document
TreeNodeShape2
ABCD
Ring
Arrow
TapeShape1
ManualShape1
TreeNodeShape1
TitleShape1
Ring
TreeNodeShape2
ABCD
Document to store at the server
Arrow
TreeNodeShape1 TreeNodeShape1 63,15 236,91
Konsole (Konsol... Konso... Kylix - .. (file:/t... Mess... Project... Object... Unit1.pas Form1 Editing...
One Two
Mon 20:11 Apr 02

want is to introduce more bugs when performing a comparative test of two setups.

The result? Success! On my 200MHz machine, the Kylix IDE was now a much happier bunny than it had been before. I could type as fast as I wanted and the text editor kept up with me. Tooltip hint windows in the Component Palette now worked properly and the whole thing was a lot snappier than before. Maybe not as snappy as Delphi running under Windows, I'll grant you, but at least it was jogging on two feet instead of crawling on four... Definitely usable now.

So does this all mean that VMware was the big culprit here? Before cogitating on that particular issue, I'd like to touch on the question of application performance.

Performance Issues: Getting Your Threads In A Knot?

One of my favourite Delphi projects has always been the THRDDEMO application which you can find in the THREADS subdirectory where all the other demo projects are stored. I've always liked this little application for two reasons: firstly, it gloriously demonstrates the superiority of the QuickSort algorithm, and secondly it's a nice little demonstration of the power of threading. Each of the three sort algorithms continues on its merry way, unaware that it's sharing the processor with a couple of its cousins.

When I first tried out the thread demo using my Pentium III machine under VMware, I was horrified by the apparent performance of the application. Something that took just a couple of seconds under Delphi and Windows was now taking an eternity. Once again, something was obviously very wrong.

I decided to look into this issue further and measured the performance of THRDDEMO on all three available platforms: Linux with VMware, native Linux, and of course good old Delphi under Windows 2000. The results are very interesting as you can see from Table 1. The centre column indicates the amount of time required for the QuickSort algorithm to finish sorting while the rightmost column gives the time required for all three sort algorithms to finish.

I have normalised the figures for the middle row (the case where we're running SuSE 7.0 natively) to allow for the fact that this was done on a 300MHz machine whereas the other two cases were measured on my 500MHz PC. For sure, I accept that there isn't a linear relationship between clock frequency and performance, a 1GHz machine won't run programs twice as fast as a 500MHz machine, for example, but any errors in my correction are insignificant compared to the huge difference in timings that are shown here. And yes, I did check these figures several times, and took an average for each case.

Whichever way you look at it, these results are extraordinary. A program that takes less than two seconds to execute under Windows takes no less than 47 seconds under a native Linux configuration. Moreover, it takes over five minutes to run when executing under VMware.

Platform	QuickSort	Time To Finish All Sorts
SuSE 7.0 (VMware)	2 minutes, 5 secs	5 minutes, 13 seconds
SuSE 7.0 (native)	7 seconds	47 seconds
Windows 2000 (with Delphi)	Not measurable	Less than 2 seconds

➤ *Table 1*

I decided to sit down and consider my options. What could be causing this huge performance problem? Could it be VMware? I really didn't think so: by the very nature of the beast, there are obviously overheads in VMware, but not *that* amount of overhead! If the problem *was* VMware, then it was clearly being caused by a combination of VMware and something else.

What about threads? Was it possible that the Linux thread dispatching mechanism was hopelessly inefficient? This would fit with the fact that I was seeing such appallingly bad performance from the THRDDEMO program but, on the other hand, if Linux threads were broken, then surely the whole operating system would run like treacle, and that's certainly not the case. Hmmm... I was running out of options. How about Qt? Again, this seemed highly unlikely. I know for a fact that Qt is a very efficient, professionally written, C++ class library that (in my experience) leaves many other class libraries in the dust from a performance point of view.

It suddenly struck me that I'd conducted all these measurements while running THRDDEMO from inside the IDE itself. How would things look if I were to run the program apart from the IDE. Silly idea? Shouldn't make a difference, right? Well, guess what...

When I tried running the thread demo program from the Linux command line, the above timings plummeted to no more than two seconds for the QuickSort algorithm to complete, with around 6-8 seconds for all three sort algorithms to run to completion. These figures were very similar for both variations of Linux, (ie running

➤ *Listing 1*

```
procedure TThreadSortForm.StartBtnClick(Sender: TObject);
begin
  RandomizeArrays;
  StartBtn.Enabled := False;
  BubbleSortBox.Invalidate;
  with TBubbleSort.Create(BubbleSortBox, BubbleSortArray) do
    Execute;
  SelectionSortBox.Invalidate;
  with TSelectionSort.Create(SelectionSortBox, SelectionSortArray) do
    Execute;
  QuickSortBox.Invalidate;
  with TQuickSort.Create(QuickSortBox, QuickSortArray) do
    Execute;
  StartBtn.Enabled := True;
  ArraysRandom := False;
end;
```

native and running under VMware) and were pretty much in line with what I'd have expected to see in the first place.

Old Wine In New Wineskins?

The bottom line, then, is that if you run certain types of application from the IDE, you'll encounter a massive drop in performance. I wasn't sure whether this was generally true, or whether it was only the case when dealing with multi-threaded applications. My gut feeling was that the thread switching was the real source of the problem, but I couldn't be sure. Accordingly, in order to investigate this further, I made some modifications to the THRDDEMO program. In the file SORTTHDS.PAS, I changed the `TBubbleSort`, `TSelectionSort` and `TQuickSort` classes so that they were derived from `TObject` instead of `TThread`. This necessitated some other minor changes which should be pretty obvious if you try the same thing for yourself. In the main THSORT.PAS file, I massaged the `StartBtnClick` routine so as to look like Listing 1.

The whole idea, of course, is to re-jig the program so that it runs the three sort algorithms sequentially, one after another, without using threading. Once again, this gave me a new set of timings. Under VMware and executing from within the IDE, the QuickSort algorithm was now executing so quickly as to be impossible to time accurately with my stopwatch (compare this to a time of over 2 minutes using threading!) and all three sort algorithms had finished executing in just over four seconds.

At this point, I could have hacked the code around some more, so as to programmatically perform its own timings, but I didn't see any point. From my perspective, I was now confident that the problem only applies to multi-threaded applications running inside the IDE, and I was equally confident that the use of VMware somehow seems to exacerbate the problem. Whether or not this is going to be an issue for you obviously depends on the type

Deployment Issues And The GPL 'Virus'

The deployment of Kylix-authored executables is obviously an important consideration. If you check the DEPLOY file on the Kylix CD, you'll find this: *"With Kylix, there is no runtime interpreter. You need only deploy your application and any required packages (BPLs). For simple applications you can distribute a standalone executable."*

Unfortunately, this is not entirely true. Assume for a moment that you've created a do-nothing, application containing only a single form with no controls on it. A non-packaged version of this executable will weigh in at around 380Kb, falling to less than 20Kb when using packages. If you take the non-packaged executable and try running it on a virgin Linux system (ie a system which hasn't had Kylix installed) then the program definitely won't run. Assuming that everything else is OK (compatible versions of glibc, libjpeg, Qt, etc), your app still won't run because it needs a shared library called LIBQTINTF.SO.

As you'll recall from previous discussions, CLX sits on top of Qt. Qt, however is a C++ class library and it can't therefore be called directly from Kylix. (That said, I'd still love to see a general purpose mechanism for calling C++ shared libraries in Kylix 2.0!) LIBQTINTF.SO essentially acts as the 'glue' between CLX and Qt; the library presents a 'flat', procedural interface to CLX, but internally it converts everything into a true object oriented call on the 'real' Qt code which is located inside `LIBQT.SO`. Thus, if you peek inside LIBQTINTF.SO, you'll find it exporting a 'flat' routine called `QTimer_start`. Internally, this is resolved onto a call to `QTimer:start`.

Forgetting about versioning issues, LIBQT.SO will almost certainly be present on a virgin Linux installation, but the same isn't true of `LIBQTINTF.SO`. Because this is a specialised glue library, it's specific to Kylix. Therefore, you will need to deploy this 1.5Mb file as part of your product distribution. Obviously, if you create a packaged application, then the needed packages will have to be deployed as well.

Hint: The simplest way of figuring out what packages and shared libraries are needed by a particular executable is to use the standard Linux utility `ldd`. Just type `ldd` on the command line followed by the name of the executable you're interested in and you'll be presented with a list of all the libraries and packages needed by that file. You can also run `ldd` against a library/package to determine what libraries it needs, in turn. It's rather like a command-line version of Microsoft's DEPENDS utility, but not so fancy. As is the case with Windows, this utility obviously can't 'see' any library references that are made at runtime, using `LoadLibrary`, `GetProcAddress`, etc.

It's to be hoped that all the Linux vendors will include the standard Kylix packages and shared libraries (such as LIBQTINTF.SO) in future distributions, this would greatly reduce the deployment size of Kylix-authored software. However, this is by no means certain; you don't need me to tell you that the Open Source brigade frequently view commercial software with a very jaundiced eye. Perhaps this will change once the Open Edition of Kylix is released, but don't hold your breath...

This brings me on to the thorny issue of the GPL. I've been ticked off by Hallvard Vassbotn and others for blurring the distinction between GPL software on the one hand, and freeware on the other, so I'll try and be clear here! As you know, the Open Edition of Kylix won't itself be GPL software (you won't get the source code, more's the pity!) but it can only be used to create GPL products.

Some folks have likened GPL to a very infectious virus, the high tech equivalent of foot and mouth! Take a reusable component that's been released under the GPL, incorporate it into a project you're developing and hey-presto, suddenly the entire project has got to be released under GPL too. The ramifications of this are clear: if you're building a commercial, closed-source application using one of the commercial variants of Kylix, it is simply not possible for you to use a GPL component that has been created by someone with the Open Edition. This will inevitably lead to a fragmentation of the Delphi/Kylix component 'scene' with some controls being released under GPL, and some not. I very much hope that Borland change their stance on this and allow the Open Edition of Kylix to be used to create any type of non-commercial application, without making recourse to the GPL.

of application you're developing. But I'd suggest that multi-threading encompasses quite a large class of prospective applications, and all the more so if you choose Indy for internet programming. As pointed out earlier, the blocking nature of these components means that the use of threads is pretty well mandatory.

I'm moving into the area of speculation now, but I am more than a little suspicious that the woeful performance of the Kylix IDE under VMware is essentially another manifestation of the same problem. After all, the Kylix IDE is itself a Kylix application, and, yes, it also uses threading to handle things like code completion. Could it be that the glacial speed of THRDDEMO running in the IDE is in some way related to the IDE's inability to cope with anything more than one keystroke per second? Personally, my hunch is that the two relate to the same problem.

And did I mention Wine? As you will know from my past writings, Borland took the decision to use Wine (www.winehq.com) in the development of the Kylix IDE. This was done primarily for the sake of expediency, because both the VCL library and the existing Delphi IDE make use of Windows API calls. By using Wine, they were able to get the IDE up and running that much faster without having to convert existing Win32 code for Linux. This means that, internally, the Kylix IDE is almost certainly linked with some variant of VCL rather than CLX. For similar reasons (as I mentioned in my first-look piece) Kylix has the same dockable IDE windows that we have in Delphi, whereas applications developed with Kylix have no docking window support at all.

At this point, the purists will undoubtedly step in and point out that the Kylix IDE is implemented using Winelib, and not Wine. This is certainly true, but so what? Strictly speaking, Wine is an *emulator* which allows unmodified Windows executables to run under Linux. Winelib, on the other hand, is a library of routines which implement the Windows API under Linux. By linking against Winelib, a native Linux application can make Win32 calls. Either way, it's the same code that ends up getting called in both cases since the Wine documentation clearly states that:

Most of Winelib's code consists of the Win32 API implementation. Fortunately this part is 100 percent shared with Wine. The remainder consists of Windows compatible headers and tools like the resource compiler (and even these are used when compiling Wine).

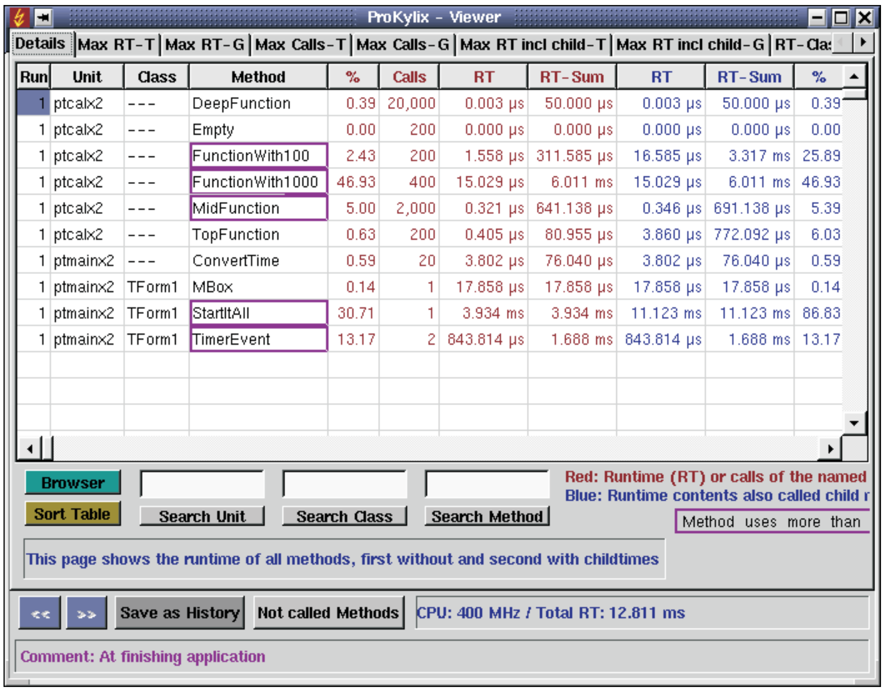
It's my suspicion (and again, this is only speculation on my part, I am not an expert with Wine or Winelib) is that there is something wrong with the Wine code that implements threads under Linux. If I'm wrong, I'm sure someone won't hesitate to tell me (!) but I just hope that these performance problems will disappear with the next release of Kylix, which is to be (we understand) 100% native with no reliance on Winelib.

The Plain Man's Guide To Running Kylix Programs

The main Kylix CD includes four important text files named README, DEPLOY, INSTALL and PREINSTALL. You should take care to read all these files, though not necessarily in that order! Borland's intention was that this information would get Linux novices up to speed as quickly as possible. From where I'm sitting, this hasn't been a total success. Judging from the traffic in the Kylix newsgroups, there's one issue which new Kylix users struggle with more than anything, and that's the thorny question of how to run a Kylix application outside of the IDE. Not only do the aforementioned files make assumptions about the Linux knowledge of their readers, but some newsgroup participants (who should know better) have displayed a rather patronising attitude which has only fanned the flames of discontent. If I refer to an 'RTFM attitude', I'm sure you will understand what I mean.

So here, for Linux newbies everywhere, is a plain and simple explanation of how to configure your system to run Kylix executables. Firstly, it's important to appreciate that, as with Delphi, you can create two types of application which I generally refer to as packaged and non-packaged applications. Just as with Delphi, the `Project|Options` dialog has a

➤ *Figure 4: ProKylix is probably the first available Kylix application profiler that I've seen. With a claimed timer resolution that's measured in microseconds, it's something that we'll be taking a closer look at in due course.*



Run	Unit	Class	Method	%	Calls	RT	RT-Sum	RT	RT-Sum	%
1	ptcalx2	---	DeepFunction	0.39	20,000	0.003 µs	50.000 µs	0.003 µs	50.000 µs	0.39
1	ptcalx2	---	Empty	0.00	200	0.000 µs	0.000 µs	0.000 µs	0.000 µs	0.00
1	ptcalx2	---	FunctionWith100	2.43	200	1.558 µs	311.585 µs	16.585 µs	3.317 ms	25.89
1	ptcalx2	---	FunctionWith1000	46.93	400	15.029 µs	6.011 ms	15.029 µs	6.011 ms	46.93
1	ptcalx2	---	MidFunction	5.00	2,000	0.321 µs	641.138 µs	0.346 µs	691.138 µs	5.39
1	ptcalx2	---	TopFunction	0.63	200	0.405 µs	80.955 µs	3.860 µs	772.092 µs	6.03
1	ptmainx2	---	ConvertTime	0.59	20	3.802 µs	76.040 µs	3.802 µs	76.040 µs	0.59
1	ptmainx2	TForm1	MBox	0.14	1	17.858 µs	17.858 µs	17.858 µs	17.858 µs	0.14
1	ptmainx2	TForm1	StartItAll	30.71	1	3.934 ms	3.934 ms	11.123 ms	11.123 ms	86.83
1	ptmainx2	TForm1	TimerEvent	13.17	2	843.814 µs	1.688 ms	843.814 µs	1.688 ms	13.17

Red: Runtime (RT) or calls of the named
Blue: Runtime contents also called child r
Method uses more than
This page shows the runtime of all methods, first without and second with childtimes
CPU: 400 MHz / Total RT: 12.811 ms
Comment: At finishing application

checkbox marked `Build with runtime packages` which allows you to control whether or not the various runtime packages are needed by your application.

You might be forgiven for thinking that a non-packaged application is completely standalone, but this isn't the case. As we all know, the CLX framework is sat on top of the Qt C++ library, and all Kylix applications need to find this shared library. (In Linux-land, DLLs are referred to as shared libraries.) If you simply install Kylix, fire up the IDE, create a do-nothing program with an empty window and then try running the executable from the command-line, you'll be told that the system can't find a library called libqtintf, or words to that effect.

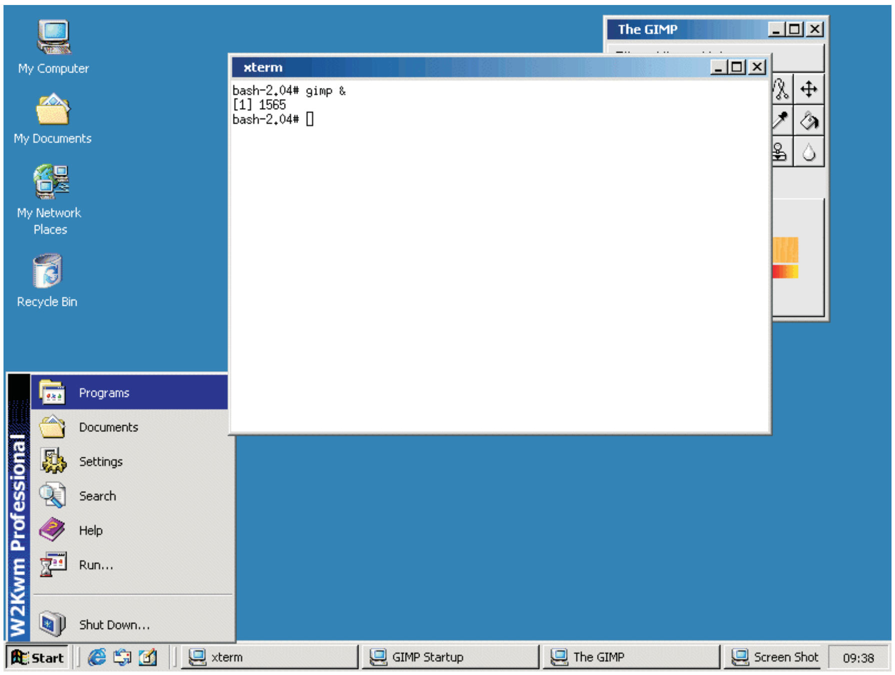
As with DOS, Windows, etc, Linux has its equivalent of the venerable `PATH` environment variable, and another, `LD_LIBRARY_PATH`, which is more concerned with shared libraries. Unless these are set up correctly, you won't be able to run your Kylix apps outside of the IDE. The IDE, of course, knows where all these goodies are located and establishes its own runtime environment for applications run from within itself.

Borland provide a shell script (like a batch file!) called kylixpath, which is automatically massaged at install-time according to where you decided to install Kylix. Let's assume for the sake of argument that you told the installer to put Kylix in /usr/local/kylix.

By default, the install program offers to put Kylix into a directory below the home directory of the current user. Although installing Kylix from the root account is a good idea, habitually running Kylix from root is a bad idea, for reasons that should be obvious. Moreover, if you install Kylix below the /root directory, it will be inaccessible to ordinary users. In my opinion, the install program should detect that it's being executed by root and change the default install directory to /usr/local/kylix or something similar, or at least warn of the consequences of installing below /root. This is the single most glaring deficiency in the Kylix install program.

Assuming Kylix *is* installed into /usr/local/kylix, then the kylixpath script will be located at /usr/local/kylix/bin/kylixpath. Although you could manually execute this script every time you start a new session, that would be very tedious. Assuming you're using the BASH shell, a much better idea is to place the necessary shell command into a special file called ~/.bashrc so that it's executed automatically every time you start a new shell. Yes, that's right, the Linux equivalent of AUTOEXEC.BAT!

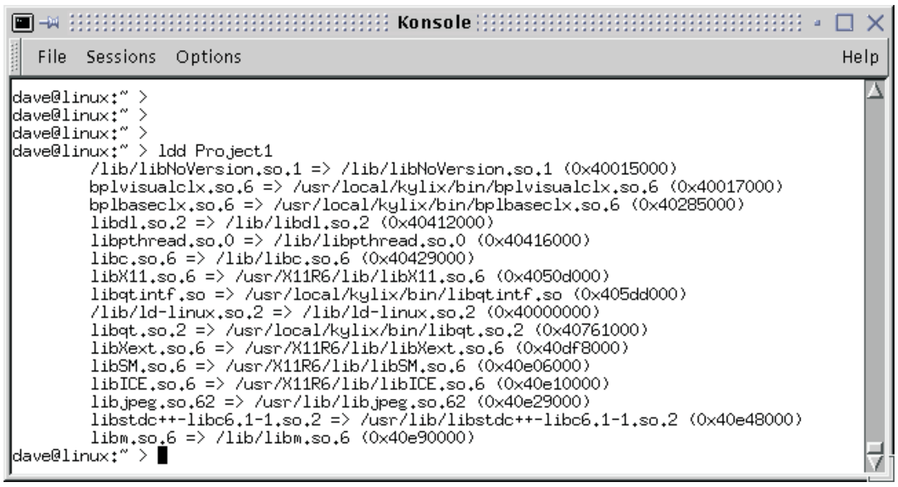
What's not obvious from the foregoing is that the tilde (~) is shell-speak for your home directory. If your username is `dave`, then this will be /home/dave. So, the procedure is, go to your home directory, locate the file .bashrc and... what do you mean, you can't see any file by that name? Another Linux convention is that files that begin with a period are normally invisible to file managers and the like. Just open your favourite file



```
bash-2.04# gimp &
[1] 1565
bash-2.04# 
```

➤ *Figure 5: It looks like Windows, but this is W2KWM, the first Kylix-authored window manager for Linux! Find it, free with source code, at www.vclcrawler.com/w2kwm/.*

➤ *Figure 6: ldd is a useful Linux utility which tells you what shared libraries and packages are used by an executable. Use it when determining how to deploy a Linux application.*



```
dave@linux:~ >
dave@linux:~ >
dave@linux:~ >
dave@linux:~ > ldd Project1
        /lib/libNoVersion.so.1 => /lib/libNoVersion.so.1 (0x40015000)
        bplvisualclx.so.6 => /usr/local/kylix/bin/bplvisualclx.so.6 (0x40017000)
        bplbaseclx.so.6 => /usr/local/kylix/bin/bplbaseclx.so.6 (0x40285000)
        libdl.so.2 => /lib/libdl.so.2 (0x40412000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40416000)
        libc.so.6 => /lib/libc.so.6 (0x40429000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x4050d000)
        libqtintf.so => /usr/local/kylix/bin/libqtintf.so (0x405dd000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
        libqt.so.2 => /usr/local/kylix/bin/libqt.so.2 (0x40761000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x40df8000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x40e06000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x40e10000)
        libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x40e29000)
        libstdc++-libc6.1-1.so.2 => /usr/lib/libstdc++-libc6.1-1.so.2 (0x40e48000)
        libm.so.6 => /lib/libm.so.6 (0x40e90000)
dave@linux:~ > █
```

manager, such as kfm, navigate to your home directory and select `Show Hidden Files` from the `View` menu. At this point, .bashrc should appear and you can edit it in the usual way.

Well actually, you can't. The bad news is that, by default, ordinary users don't have write access to this file so, here again, we have another Linux assumption that hasn't been spelt out for you! It's probably easiest to just edit .bashrc immediately after installing Kylix while you're still logged on as root.

Assuming you installed Kylix into the directory mentioned earlier, it's now just a case of appending the following line to the end of dave's .bashrc and saving the file:

```
source /usr/local/kylix/bin/
  kylixpath >/dev/null
```

In plain English, this shell command says 'read and execute all the shell commands in the kylixpath script, and throw away any console output that gets generated'.

With all that done, log in as `dave`, and you should now be able to simply execute your Kylix executables from the command line. That was easy, wasn't it? But in all seriousness, I think Borland need to try a lot harder if they expect Windows developers to persevere with this stuff. Those tersely-worded README files on the CD are jam-packed full of assumptions about the Linux knowledge of the end-user, and they really need padding out with a lot more background information.

Conclusions

Some folks will doubtless accuse me of having spent too long looking at performance issues relating to the execution of applications from within the Kylix IDE. That's a fair accusation, but having got the bit between my teeth, I found it hard to let go until I'd found that multi-threaded applications were the cause of the problem!

I'm anxious that you shouldn't come away with the impression that this is a negative article: it definitely isn't. Kylix is undoubtedly the best software development environment for Linux at the present time, but a certain amount of preparation is needed to get the best out of it, and this could have been helped with better documentation from Borland.

I'll briefly summarise by saying that, first, you should avoid running under VMware if that is at all possible. I'm a great fan of VMware, but there seems to be some sort of thread-switching problem in the IDE which is exacerbated by the presence of VMware. Hopefully this will be fixed soon, but in any event you'll get better performance, and have more memory available, by 'going native'.

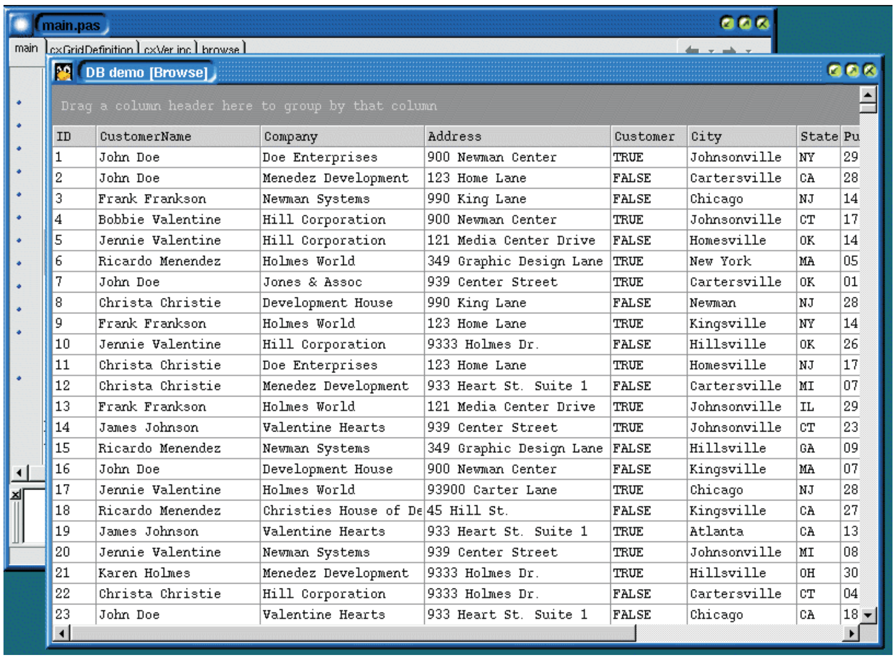
Second, if you're not that familiar with Linux, follow my *Plain Man's Guide* to setting up your .bashrc file such that you can run Kylix applications outside the IDE. If you still have no joy, ask in one of the Kylix-specific newsgroups I've alluded to earlier.

In future issues, we'll be looking at topics such as Delphi/VCL compatibility: what are the main things to watch out for when porting your application from Windows to Linux or vice versa? Have Borland done a good job of wrapping the underlying Qt controls so as to make them behave, as far as possible, like the common controls present on Windows? Stay tuned for a lot more Kylix coverage in the future, starting with Brian's article on Apache shared modules using Kylix, and Bob Swart's coverage of dbExpress, in this very issue.

So, to answer my original question, is Kylix worth buying? Absolutely. As you'd expect from a '1.0' product, it has its rough edges, and I anticipate that the IDE will be a great deal smoother and more responsive once the Wine has been put back in the bottle, so to speak! I'm not overwhelmed by the number of third-party components that are available, but I'm very impressed by the quality of some of the components that are. If you want to get in on the ground floor of the Kylix revolution, now is undoubtedly the time to do it.

Dave Jewell is the Technical Editor of *The Delphi Magazine*, contact him at TechEditor@itecuk.com

➤ *Figure 7: At time of writing, Developer Express are keeping fairly quiet about their plan for Kylix, but it's clear that they have ported at least some of their controls across to CLX...*



ID	CustomerName	Company	Address	Customer	City	State	Pu
1	John Doe	Doe Enterprises	900 Newman Center	TRUE	Johnsonville	NY	29
2	John Doe	Menedez Development	123 Home Lane	FALSE	Cartersville	CA	28
3	Frank Frankson	Newman Systems	990 King Lane	FALSE	Chicago	NJ	14
4	Bobbie Valentine	Hill Corporation	900 Newman Center	TRUE	Johnsonville	CT	17
5	Jennie Valentine	Hill Corporation	121 Media Center Drive	FALSE	Homesville	OK	14
6	Ricardo Menendez	Holmes World	349 Graphic Design Lane	TRUE	New York	MA	05
7	John Doe	Jones & Assoc	939 Center Street	TRUE	Cartersville	OK	01
8	Christa Christie	Development House	990 King Lane	FALSE	Newman	NJ	28
9	Frank Frankson	Holmes World	123 Home Lane	TRUE	Kingsville	NY	14
10	Jennie Valentine	Hill Corporation	9333 Holmes Dr.	FALSE	Hillsville	OK	26
11	Christa Christie	Doe Enterprises	123 Home Lane	TRUE	Homesville	NJ	17
12	Christa Christie	Menedez Development	933 Heart St. Suite 1	FALSE	Cartersville	MI	07
13	Frank Frankson	Holmes World	121 Media Center Drive	TRUE	Johnsonville	IL	29
14	James Johnson	Valentine Hearts	939 Center Street	TRUE	Johnsonville	CT	23
15	Ricardo Menendez	Newman Systems	349 Graphic Design Lane	FALSE	Hillsville	GA	09
16	John Doe	Development House	900 Newman Center	FALSE	Kingsville	MA	07
17	Jennie Valentine	Holmes World	93900 Carter Lane	TRUE	Chicago	NJ	28
18	Ricardo Menendez	Christies House of De	45 Hill St.	FALSE	Kingsville	CA	27
19	James Johnson	Valentine Hearts	933 Heart St. Suite 1	TRUE	Atlanta	CA	13
20	Jennie Valentine	Newman Systems	939 Center Street	TRUE	Johnsonville	MI	08
21	Karen Holmes	Menedez Development	9333 Holmes Dr.	TRUE	Hillsville	OH	30
22	Christa Christie	Hill Corporation	9333 Holmes Dr.	FALSE	Cartersville	CT	04
23	John Doe	Valentine Hearts	933 Heart St. Suite 1	FALSE	Chicago	CA	18