第 16 卷 第 4 期 郑州轻工业学院学报(自然科学版) Vol. 16 No. 4
2001 年 12 月 JOURNAL OF ZHENGZHOU INSTITUTE OF LIGHT INDUSTRY (Natural Science) Dec. 2001

文章编号:1004 - 1478(2001)04 - 0031 - 04

基于 Kylix 的 CLX 跨平台编程

卢中宁[1], 李金达[2], 康国磊[3], 李迎昕[4]

(1. 郑州轻工业学院 计算机科学与工程系,河南 郑州 450002;
2. 郑州市电信局,河南 郑州 450052;
3. 郑州轻工业学院 现代教育技术中心,河南 郑州 450002;
4. 一拖国际经济贸易有限公司,河南 洛阳 471003)

摘要:为了实现 Windows 与 Linux 下的跨平台编程,需要引入基于 Kylix 的交叉平台控件库(CLX). CLX 先将应用程序的源代码在 Windows 或 Linux 下面编译,再转移到 Linux 或 Windows 下进行重新编译,通过建立一个接口层来获得 Qt 的类,并还原成 C 函数,最后把它们包装成 Windows 下的 DLL 或是 Linux 下的共享对象. CLX 可方便地移植 Windows 或 Linux 平台下的 Dephi/C++Builder 程序.

关键词:程序;数据库;交叉平台控件库
中图分类号:TP311.52 **文献标识码**:A

0 引言

Visual Basic 的出现,使程序的编制变得简单,从而让程序员摆脱了复杂的 GUI 及内存管理和任务管理,使他们可以专心于本职的工作. 对于那些没有时间记忆 Windows API 的程序员来说,VB 是一个好工具. 后来,Borland 发布了 Delphi,它更高效、更易于使用,具有更高性能,是真正的 OOP,要开发 Windows 程序,Delphi 就是最好的选择. 但由于 Delphi 出现太晚,Windows 程序员们已经选择了其他工具,比如 VB,Smalltalk 或者 C++及某个类库. 到目前为止还没有基于 Linux 的组件化、全编译、纯 OOP 的 RAD 工具,而 Kylix 就是 Linux 版的 Visual Basic,Linux 平台下的第一个 RAD 工具.

Kylix 项目是一个基于 Linux 的双向可视化组件开发环境,可以开发 GUI,Internet,数据库和服务器应用程序. Kylix 项目包括一个新的高速的用原生代码编写的基于 Linux 的 C/C++/Delphi 编译器,还包括一个 Linux版的 Borland VCL 类库. 这个 Linux 版的 VCL 既能简化 Linux 应用程序的开发,又便于移植 Windows 和 Linux 平台下的 Delphi/C++ Builder 程序.

1 交叉平台控件库(CLX)

我们可以在 Windows 下写一个应用程序,然后把源代码转移到 Linux 下面重新编译——反之亦然. 这个新的 VCL 叫做 CLX,意即"交叉平台控件库(component library cross-platform)". CLX 包含整个随 Kylix 发布的交叉平台库,它可以分为下面 4 个子类:

1) BaseCLX 就是 RTL,包含并且升级了 Classes.pas;
2) VisualCLX 包含了用户界面类,比如常用的控件;
3) NetCLX 包含 Internet 部分,比如 Apache 等;

收稿日期:2001 - 07 - 02
作者简介:卢中宁(1974—),男,河南省洛阳市人,郑州轻工业学院助教,主要从事计算机网络和数据库研究.

4) DataCLX 包含交叉平台的数据库控件.

从本质上说,VisualCLX 就是我们熟悉的 VCL,只不过还包含比"可视构件"更多的东西,在此,我们只谈"可视"控件. 类似 Button, Edit, ListBox, PageControl, StatusBar, ProgressBar 等的控件,都可以在交叉平台下重新实现. 但是,现在的 VCL 非常依赖 Windows 平台,要实现跨平台编程需在剥离了所有的 Windows 元素之后,把它们用别的工具包代替. 在 Linux 下,有被称作 Widgets 的大量工具包,它包含标准 Windows 控件(如 Buttons). Qt 是非常流行的 Widgets, Qt 与 CLX 的关系就好像 Windows API/通用控件与 VCL 的关系. 对于 Linux 下的 Delphi 的定制构件开发者来说,Qt 有一些明显的好处:

1) 它是一个广泛使用的 Linux 下的 Widgets 集,被流行的 KDE 桌面采用;

2) 它引入大量标准 Widgets,并且具有消息循环;

3) 它的类看上去非常像 VCL 控件;

4) 它的开发与 Windows API 风格非常相似;

5) 它的图形模块与 VCL 的图形模块相似.

Kylix 不仅支持 KDE,而且还支持 Gnome,所以它能在其他桌面上执行. Kylix 的目标是让开发者方便地将应用程序转移到 Linux 下,转移后只有一些控件的少数属性被去掉了,增加了一些新的属性,大部分新旧控件的名字和属性都是一样的,可以平稳地转移应用程序. 不过应该注意到 CLX 类的名字都被加上了一个前缀 Q,比如 StdCtrls 变成了 QStdCtrls,有些类被稍微搅乱了一点,在类继承上有一些细微差别(见图 1).

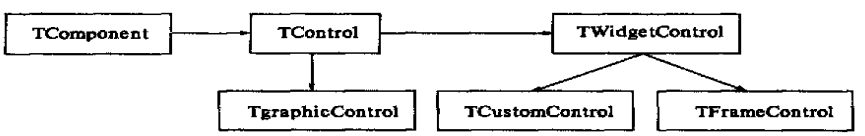

图 1 在类继承上面的细微区别

CLX 的这个前缀 Q 不一定是最终版本的前缀. TWinControl 现在变成了 TWidgetControl,这是为了减少影响,为 TWinControl 添加了一个 TWildgetControl 的别名. TWidgetControl 和它的后代都有一个 Handle 属性,隐式地指向 Qt 对象. 另外,有一个 Hooks 属性指向一个 Hook 对象,用来实现事件机制. OwnerDraw 将被一种叫作 Styles 的方法替代, Styles 是 Widgets 或应用程序显示新面孔的一种机制,类似于 Windows 下面的贴图(Skins). 不过,新旧控件中有一些还是一样的,比如 TCanvas(包括 Pens,Brushes 等).

2 CLX 的工作过程

Qt 是一个 C++ 的工具集,所以 Widgets 是 C++ 对象,另一方面,CLX 是用 Object Pascal 写的,但 Object Pascal 不能直接和 C++ 对象对话. 因为 Qt 在几个地方都使用了多继承,所以可以建立一个接口层(interface layer)来获得所有 Qt 的类,并把它们还原成一系列普通的 C 函数,最后把它们包装成 Windows 下的 DLL 或是 Linux 下的共享对象.

每个 TWidgetControl 都有 CreateWidget, InitWidget 和 HookEvents 虚方法,并且几乎总是被重载. CreateWidget 创建 Qt 的 Widgets,然后指派 Handle 到 FHandle 这个私有域变量中. 当 Widgets 被构造后, InitWidget 被调用,然后 Handle 有效. 一些属性赋值将从 Create 这个构造函数转移到 InitWidget,这将能够做到延迟构造一个对象,直到真的需要它的时候. 例如,有一个属性叫 Color,在 SetColor 里面,可以通过 HandleAllocated 来检测是否有一个 Qt 的 Handle. 如果 Handle 已经分配,就可以正确地调用 Qt 来设置颜色;如果没有分配,可以把值保存在一个私有域变量中,然后在 InitWidget 中设置属性. HookEvents 是一个虚方法,它钩住 CLX 控件的事件方法到一个特殊的 Hook 对象,通过这个对象与 Qt 对象通讯. 这个 Hook 对象其实是方法指针的集合. 在 CLX 的帮助下,写交叉平台控件将变得非常容易,就像你不必理解 Windows API 的复杂性而去写 VCL 控件一样,CLX 和 Qt 也是如此.

下面是一个工程文件 CalcTest.dpr. 计算器控件运行在 Windows 下和 Linux 下就很像标准的 Microsoft

Windows 计算器.

```
program CalcTest;
uses
  SysUtils, Classes, QControls, QForms, QStd-
Ctrls, Qt,
  QComCtrls, QCalc, Types;
type
  TTestForm = class(TForm)
    Calc: TCalculator;
  public
    constructor Create(AOwner: TComponent);
    override;
  end;
var
  TestForm: TTestForm;
  {TTestForm}
constructor TtestForm.Create(AOwner: TCompo-
nent);
begin
  inherited CreateNew(AOwner);
  SetBounds(10,100,640,480);
  Calc := Tcalculator.Create(Self);
  // Don't forget: we have to set the parent.
  Calc.Parent := Self;
  Calc.Top := 100;
  Calc.Left := 200;
  // Uncomment these to try other Border effects:
  // Calc.BorderStyle := bsEtched;
end;
begin
  Application := Tapplication.Create(nil);
  Application.CreateForm(TTestForm, TestForm);
  TestForm.Show;
  Application.Run;
end.
```

在这个控件中有 2 个重要的方法.BuildCalc 创建所有的按钮,并且把它们摆放到正确的位置.它使用了一个叫 TButtonType 的枚举类型来控制按钮的"功能",还有少量的信息作为整型保存在 Tag 属性里面;另外,所有的计算器按钮保存在一个叫作 Btns 的受保护的记录数组里面,类型是 TButtonRecord.

```
TButtonRecord = record
  Top: Integer;
  Left: Integer;
  Width: Integer;
  Height: Integer;
  Caption: string;
  Color: TColor;
end;
```

这样做能够较容易地在一个循环里面设置所有的按钮,而不用写一大串的 Tbutton.Create 调用.注意所有按钮的 OnClick 句柄都指派给了 TCalculator 的 Calc 方法,直接指派到一个自定义事件是不错的,因为所有按钮都在计算器的内部,并且这些事件都不用被发布.

```
for i := Low(TButtonType) to High(TButton-
Type) do
with TButton.Create(Self) do
begin
  Parent := Self;
  SetBounds(Btns[i].Left, Btns[i].Top,
    Btns[i].Width,
    Caption := Btns[i].Caption;
    Color := Btns[i].Color;
    OnClick := Calc;
    Tag := Ord(i);
end;
```

另外,还有一个叫 FStatus 的 TLabel 控件.TLabel 也是 TFrameControl 的后代,若想在计算器里面使用它,可以让它具有 sunken box 的外观来显示计算器的存储记忆,就像 Windows 里面的计算器一样.Qt 标签的 Widgets 非常像 VCL 里面的 TPanel 控件.对于 CLX 里面的 TLabel,没有发布它的框架属性,但是这并不妨碍继承使用它.在 BuildCalc 里面最后要创建一个 Edit 控件来显示计算结果,计算器控件的 Text 属性和 Edit 控件的 Text 属性直接挂钩.

另一个重要的方法是 Calc,它实质上是一个庞大的 Case 语句,用来计算哪一个按钮被按下,并且决定该如何去做.如果使用了私有域变量 FCurrentValue,FLastValue 和 FRepeatValue 来保存计算的值,那么就不必再使用堆栈来实现.这个例子只是为了展示如何创建交叉平台控件,而不是如何写一个计算器.

在前面可以在 BuildCalc 中使用 Tag 属性来控制它的功能. 在这个方法里面, 将参数 Sender 强制转化成 TButton,再将它的 Tag 强制转化成 TButtonType 类型,最后赋值给 ButtonType. ButonType 就是那个 Case 语句里

面的选择器表达式.

```
ButtonType := TButtonType(TButton(Sender).Tag);
```

这些代码可以同时在 Windows 和 Linux 下面编译,而不用改动任何地方.正是仰仗于 CLX 的优点,控件全部完工,能够看到 Kylix 的一些基本效果,它有可能成为 Linux 下的 VB++.

参考文献:

[1] Charls Calvert, David Intersimone. Building Kylix Applications[M]. New York: McGraw-Hill Professional Publishing, 1997. 431—436.

[2] Eric Whipple, Rick Ross, Fletcher Bumpus. Kylix Development[M]. New York: Wordware Publishing, 1999. 293—298.

[3] 谢玉凤,姜进磊. Delphi 5 开发指南[M]. 北京:清华大学出版社,2001. 351—359.

[4] 章生立,董三立. Windows 程序设计[M]. 北京:北京航空航天大学出版社,1995. 382—387.

Programming with component library cross-platform based on Kylix

LU Zhong-ning[1], LI Jin-da[2], KANG Guo-lei[3], LI Ying-xin[4]

(1. *Dept. of Comp. Science and Eng., Zhengzhou Inst. of Light Ind., Zhengzhou* 450002, *China*;
2. *Zhengzhou Telecom Bureau, Zhengzhou* 450052, *China*;
3. *Modern Education Tech. Center, Zhengzhou Inst. of Light Ind., Zhengzhou* 450002, *China*;
4. *YTO. International, Ltd., Luoyang* 471003, *China*)

Abstract: In order to realize the bestriding between Windows and Linux in the process of programming, it needs introducing a component library cross-platform(CLX) based on Kylix. First, CLX compiles the original code of the application program under Windows or Linux, and then it compiles again after being transferring to Linux or Windows, through building a interface it can get the class of Qt and returns to the function of C, at last it will pack them into the DLL of Windows or Shared object of Linux. CLX can conveniently transplant the Dephi or C++Builder program under Windows and Linux.

Key words: program; database; component library cross-platform