光盘内附书中范例

乔林　编著

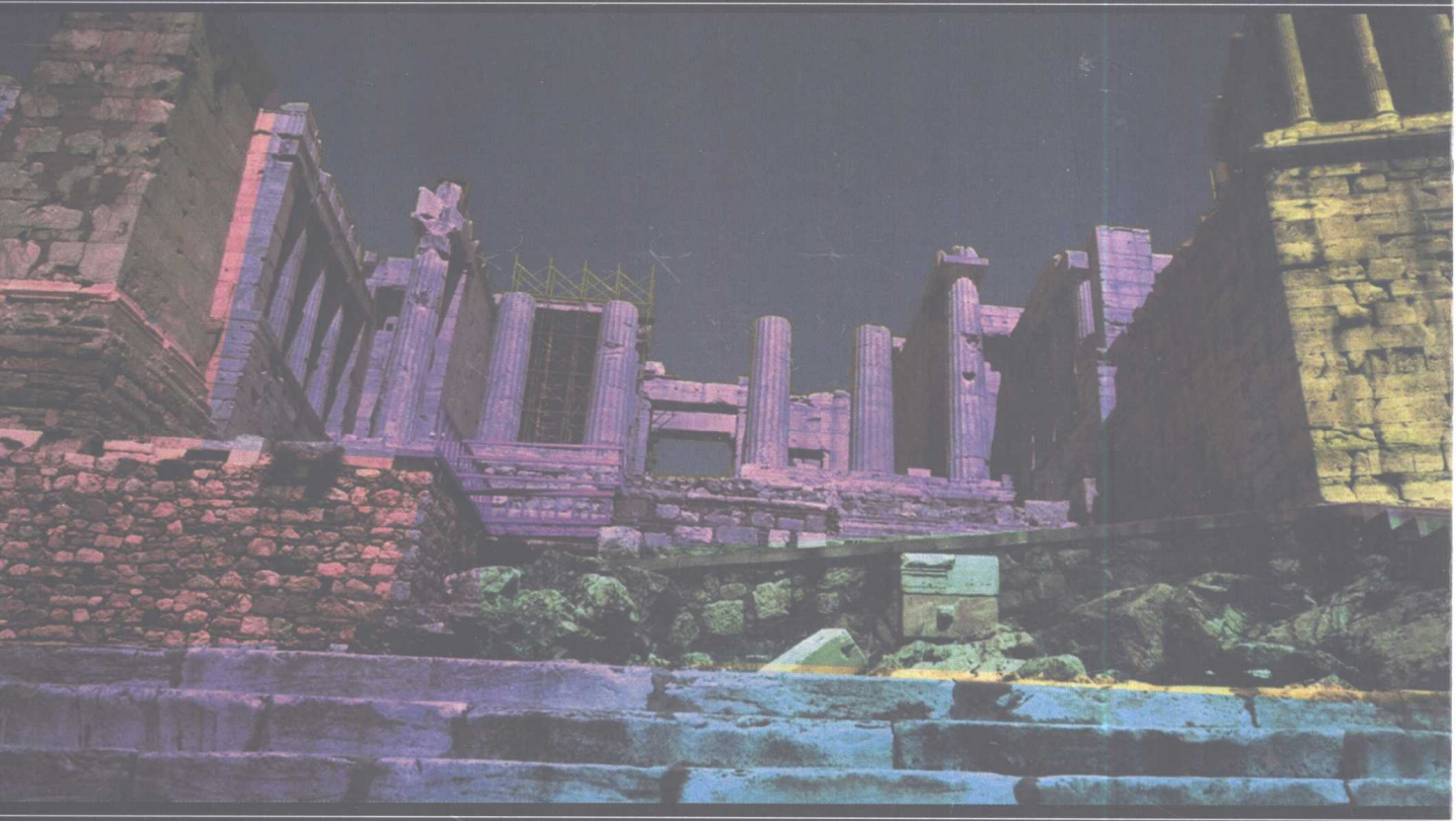
您想站在3年后软件开发的风口浪尖上吗？

您想编写既可以在Microsoft Windows下运行

也可以在Linux下运行的跨平台软件吗？

从现在起就开始学习Linux编程吧！

Kylix是您精通Linux编程的敲门砖。



Kylix程序设计

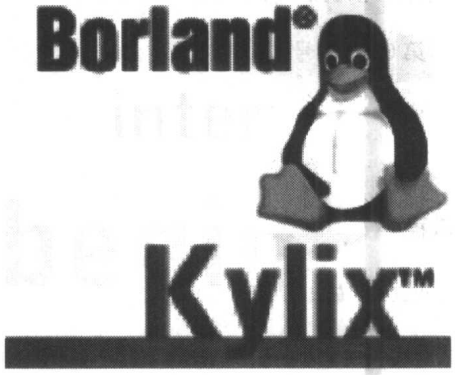
基　础　教　程

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

Kylix 程序设计

——基础教程

乔 林 编著



中 国 铁 道 出 版 社

2001 年 • 北京

（京）新登字 063 号

内 容 简 介

本书以多个应用程序实例为基础，介绍了 Kylix 程序设计的基本方法。内容涉及 Kylix 的安装与使用、Kylix 可视化开发方法、Kylix 程序结构、Kylix 的基本语言要素、Kylix 的控制结构、数组、字符串与数字、过程与函数、菜单与工具栏设计等。

书中详细剖析了各个实例，使读者学会正确的思考方法，以及如何正确的将思考方法转化为正确的程序代码。附带光盘中给出了书中所涉及的实例和练习的全部源代码。

本书是 Kylix 程序设计的入门读物，适合计算机软件开发人员和一般计算机人员，尤其是 Linux 爱好者使用。如果与本系列的其他图书配合使用效果更佳。

图书在版编目（CIP）数据

Kylix 程序设计基础教程/乔林编著.—北京：中国铁道出版社，2001.7

ISBN 7-113-04232-5

Ⅰ.K… Ⅱ. ①乔… Ⅲ.软件工具，Kylix—程序设计 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字(2001)第 035201 号

书　　名：Kylix 程序设计——基础教程
作　　者：乔　林
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：严晓舟
特邀编辑：王占清
封面设计：冯龙彬
印　　刷：北京市彩桥印刷厂
开　　本：787×1092　1/16　印张：28　字数：676 千
版　　本：2001 年 7 月第 1 版　2001 年 7 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-04232-5/TP・571
定　　价：46.00 元

前 言

可以毫不夸张地说，Kylix 是 Linux 发展的里程碑——它的出现是革命性的。

关于 Kylix

近几年来，Linux 无疑是业界上升最快的操作系统。作为最具竞争力的企业环境之一，由于无比低廉的价格，以及与价格相比实在太好的可靠性，Linux 广泛应用于 Web 应用服务器，并迅速成为 Windows 操作系统的主要竞争对手。

然而现实情况时，在 Linux 环境下开发应用程序不仅繁琐困难而且效率太低，程序员不得不进行大量的重复性劳动以编写 X Window 图形用户界面的程序代码，这种局面十分类似 Visual Basic 问世前 Windows 操作系统所面临的难局。随着 Linux 的发展，业界迫切需要一种快速开发工具，以弥补 Linux 下应用程序不足以及编程复杂的欠缺。

1999 年的 9 月 28 日，Borland/Inprise 公司正式宣布开发 Linux 环境下的快速开发工具，时隔仅一年半，Borland/Inprise 公司就适时推出了 Kylix。

Kylix 实际上是 Delphi 的 Linux 版本，它与 Delphi 一脉相承。Kylix 的出现，彻底终结了 Linux 过于艰深、曲高和寡的历史，每个普通的程序员都可以使用 Kylix 快速开发 Linux 下的应用程序。Kylix 是 Linux 发展的里程碑，它使得 Linux 第一次可以在桌面操作系统上与 Windows 一争高低。

为什么要使用 Kylix/Delphi

Kylix，或者说 Delphi for Linux，是运行于 Linux 环境下的 Delphi。无论是窗体设计器、代码编辑器还是对象检查器、部件面板，Kylix 与 Delphi 的运行界面都几乎不存在什么差别。此外，Kylix 下的编程习惯与 Delphi 几乎完全相同：键盘响应模式与鼠标操作完全一样，菜单和命令也几乎完全一样。

从集成开发环境上看，Kylix 与 Delphi 最大的不同是使用 CLX 部件库代替了 VCL 部件库。CLX 与 VCL 在众多方面都是相似的，CLX 的最大优势是 CLX 是基于 Qt 工具包，支持跨平台开发。

使用过 Delphi 的读者将会发现向 Kylix 中移植 Delphi 的代码是最容易不过的事情。除了 CLX 部件开发，大部分 Delphi 程序都只需要按照 Kylix 指出的方法做一些小改动就可以在 Linux 下直接编译运行。

您想站在3年后软件开发的风口浪尖上吗？您想编写既可以在Microsoft Windows下运行也可以在 Linux 下运行的跨平台软件吗？从现在起就开始学习 Linux 编程吧！Kylix 是您精通 Linux 编程的敲门砖。

关于本丛书

本丛书的宗旨是通过大量或深或浅的实例讲解 Kylix 的内容。整套丛书使用多个有趣的例子讨论 Object Pascal 语言的基本要素、Kylix 编程的基本与高级方法。本丛书引出的概念不是按照传统的教学顺序组织的，而是按照对读者是否有用的顺序组织的。读者会发现，这样的组织方式将使学习过程更有效。

贯穿整本丛书的是读者应该遵循什么样的步骤，应该采取什么样的思考方法，以及如何将自己的思考转化为正确的程序代码。因此，丛书中包含大量实例和练习，这些实例和练习有的难、有的易，而且在很多情况下还有一定的关联。我们希望读者在阅读本丛书的同时也上机实践。

每一章后面都包含一些问题与练习，它们的答案通常可以在书中找到线索，但也有一些练习必须通过上机实践才能完成。笔者深信，只有实践才是出真知的最佳途径。当然我们也要指出，有一些练习是稍难的。笔者同样深信，只有独立完成稍难一点点的问题才是迅速提高的唯一途径，太易的练习没有任何意义，而太难的练习则失去了实践的价值。对于较难的练习我们将给出一定的提示，请读者务必完成它们。

本丛书是如何组织的

本丛书包括四个部分：

- 《Kylix 程序设计——基础教程》：基础教程着重解决安装和使用 Kylix 面临的基本问题。考虑到 Linux 下的程序员大多对 C/C++ 很熟悉，而对 Object Pascal 语言不太了解，该书详细介绍了 Object Pascal 语言，对于初学者有极大的帮助。
- 《Kylix 程序设计——进阶教程》：进阶教程着重讨论《基础教程》遗留下来的 Object Pascal 语言的高级特性以及如何设计漂亮的应用程序界面。本书同时研究了如何使用 Kylix 开发数据库应用程序。
- 《Kylix 程序设计——实战教程》：详细讨论使用 Kylix 开发高级应用程序、多线程应用程序、Internet 应用程序、数据库应用程序、跨平台应用程序设计的基本技巧。
- 《Kylix 程序设计——类库参考手册》：提供完整的 CLX 类库参考手册，免去读者在帮助系统中跳来转去的困扰。

本丛书的内容由浅入深，没有阅读前面的内容就直接学习后面的章节可能会给读者带来一定的麻烦，当然对相应内容有相当程度了解的读者又另当别论了。

本丛书是集体劳动的结晶，参加编写的有乔林、林杜、费广正、王程铭、王志海、何隽、计莉琴、朱辰麒、乔木、蒋培、蔡宏、汪巨涛、陈爱华、王冰、乔森、徐斌、马友兰、胡一敏、李享、刘丰收、朱琨、张喜二执笔。由于编者学识水平有限，书中难免有缺点、错误和不当之处，恳请读者批评指正。

本丛书的读者对象

本丛书包含了使用 Kylix 编程的方方面面。即使读者没有任何计算机编程经验，也可以阅读本丛书。为理解本丛书的内容，读者应该熟悉 X Window 操作系统的基本知识。对读者

最低限度的假设是能够知道 X Window 的基本操作，能够知道如何使用 X Window 环境就足够了。

我们再一次地重申，本丛书的对象包含了 Kylix 的初级到高级的所有读者：

- 初学者：本丛书的《Kylix 程序设计——基础教程》和《Kylix 程序设计——进阶教程》是极好的学习材料，初学者可以从中学到几乎全部的 Object Pascal 语言细节和基本使用技巧。
- 中级读者：可以通过学习本套丛书澄清很多以前没有完全了解的概念。
- 高级读者：我们敢保证，您肯定会在本套丛书的字里行间发现很多在其他书中看不到的内容——那是作者使用 Kylix 时的独特体会，也许是本丛书最具价值的部分——这部分内容对于您理解 Borland 公司在实现 Kylix 时的设计原则有莫大的裨益。要知道，Kylix 本身是如何实现的与我们使用她时采用什么样的编程方法和编程习惯息息相关。

“边做边学”的方法是最有效的。每学完一个例子，尝试着改变一点点，或者添加一点东西，并改变一些代码将帮助读者体验进步和成功的乐趣。

编 者

2001.3

目　录

第 1 章　事件与面向对象

本章为全书的总纲。本章引出的内容大部分与 Kylix 编程没有直接的关系。我们这里之所以将其列为本书的第 1 章，是因为在使用与操作计算机时，我们时常需要面对人与计算机的辨证关系。在面向对象技术出现以来，事件与面向对象的概念与原理构成了我们对人与计算机的辨证关系的基本理解。了解这部分内容对学习 Kylix 编程有一定的理论意义，至少可以让我们少走弯路。

本章内容包括：

- 人与计算机的关系
- 用户控制
- 对象与类的基本特征

1.1　人与计算机

与其他改变人类生活的重大技术相比，计算机问世的时间不长，但它同时在很短的时间内却获得了巨大的进展。计算机的体积越来越小，速度越来越快，能力越来越强。然而机器毕竟是机器，没有人的参与和使用，再好的机器也没有任何价值。人的参与，也就是计算机操作员的参与，决定了计算机如何进行工作，以及工作的效率到底如何。不同的操作员执行计算机操作任务的效率是不同的，一个受过良好培训的操作员操作计算机的速度肯定会比没有受过培训的操作员好得多，最简单的例子莫过于经过培训的打字员的速度是笔者在录入本书时的速度所不能比拟的。

自计算机问世以来，其功能几经变迁，早已与原先的使用性质存在非常大的差异了。原先的计算机程序是一系列的记录在纸带设备上的孔洞，要输入的信息存储在纸带上。鼠标是没有的，甚至显示器也是没有的。今天的读者已经很难想象没有鼠标和显示器的计算机了。

如果程序员要使用这样的计算机进行工作，就必须像操作万吨水压机一样装料、搬动电气开关、然后下料。这类原始的操作方式延续了相当长的时间，即使在 DOS 操作系统和个人计算机广泛流行之后也顽固地生存了好几年。

我们不得不承认 DOS 操作系统的意义是不可比拟的，尽管 DOS 操作系统的用户界面在很大程度上以极差无比的交互方式工作。

- 这其实提出了一类重要的问题：随着计算机的日益普及和广泛使用，越来越多的人将接触到计算机或必须使用计算机进行工作。如果没有经过严格的培训，这些计算机操作和使用的门外汉如何才能较快地进行工作呢？计算机的应用程序正变得越来越复杂而庞大，这样的程序应该向用户或使用者提示什么样的信息以方便用户的操作呢？在这样的大环境中，如何使应用程序与用户的交互

更简单明了就是用户界面领域研究的课题。

平心而论，改善用户界面的工作是非常困难的。哪怕是一点点的进步也包含了许多杰出的研究人员和计算机专家的不懈努力。想象一下，键盘通信刚刚被用户广泛接受，显示器的使用还刚刚起步（即使使用了显示器，它们也与我们今天使用的显示器差别很大），打印机的质量不高（也许会比机械式打字机稍好一点？），使用这样的硬件设备，用户界面能好到哪去？最典型的例子莫过于：

- 计算机提示：

请输入姓名？

- 然后用户输入自己的姓名：

Qiao

- 之后计算机显示：

你好，Qiao！

（相当不错，计算机知道我的名字呢！）

- 下面有四种选择，请输入选择前面相应的数字：

（1） 开始读入数据；

（2） 进行计算并将计算结果输出到文件中；

（3） 哈哈，老板不在，让我们先玩一会；

（4） 退出系统运行。

- 你的选择是：

（居然是菜单界面！）

此时我们输入 5，系统将告诉我们输入错误请继续输入新信息。在我们输入了正确的数字之后，系统将进行下一步的工作，然后又是类似上面的界面。

上面的例子是我们假想的，请不要认为它过于幼稚，实际上有相当多的用户界面还不如它呢！像这样程序的编写总是大同小异的，好的程序会在某一点上终止运行，而坏的程序会怎么样，玩过拙劣的 DOS 游戏的读者肯定深有体会。

这样的用户界面被富有经验的程序员使用了很多年，直到计算机成为普通人的好帮手。没有多少计算机经验的用户肯定不喜欢这样难以驾御的系统，用户总是希望能够完全控制程序的运行。

1.2 用户控制

用户控制程序的概念尽管显而易见，但却要求大量的硬件和软件的技术支持。事实上，程序员往往会发现编写一个控制性差一些的程序要比编写一个控制性好的程序还要简单一些。

俗言俗语

一般来说，计算机系统的资源总是应该由操作系统控制的，但在 DOS 环境下，一旦运行某个应用程序，该应用程序将接管大多数系统资源。对用户来说，操作系统此时在干什么就不知道了。一旦应用程序出现致命的错误，结果往往只有一种可能——死机。

DOS 虽然与用户交互，但它的界面往往不太友好。在这一点上，Linux 与 DOS 完全相同。Linux 脱胎于 Minix（一种 Unix 类的小型教学型操作系统），Linux 不是为普通用户设计的操作系统，它是黑客与专家的工具。那帮“家伙”！谁也没有闲情逸致去设计友好的用户界面——对他们而言，命令行是最快的执行程序的方法。

X Window 操作系统的出现彻底改变了人们对 Linux 操作系统的印象。就像 Windows 改变了 DOS，X Window 也改变了 Linux。在 X Window 图形化的界面下，用户的操作总是以一种方便形象的方式进行，操作的结果也以显而易见的方式返回给应用程序。从这样的变革中可以看出，操作系统和应用程序一起进化了。那么我们不禁要问，是什么使这一革命性的变革成为可能的呢？

导致这种变革的原因是事件（event）。在计算机领域，我们称实际发生的某件事为事件。按一次键盘是事件；移动鼠标是事件；计算机内部时钟的每次变化也是事件。

事件对程序员意味着什么呢？假设存在这样的程序：

（1） 程序开始运行；

（2） 做某件事；

（3） 从用户处得到输入；

（4） 根据用户的输入做相应的事；

（5） 结束运行。

整个程序的运行路径是不可改变的。计算机在某一时刻等待用户的输入，用户没有输入就不能进行其他的操作。使用事件的方法重新思考上面的程序，我们有：

（1） 程序开始运行；

（2） 如果某个事件发生，则处理之；

（3） 否则继续当前的活动；

（4） 如果用户没有命令程序结束，继续监视下一个事件；

（5） 否则结束程序的运行。

在这样的程序里，程序不关心的事件即使发生也将被忽略。

1.3　对象与类的基本特征

事件是用户界面变革的基础，而对象是事件的最基本原动力。从概念上说，对象是系统中任何被观察到的实体。在构造应用系统的过程中，程序员分析问题领域，并对解决该问题所需要的部件形成观察（即产生抽象的对象表示）。我们经常说对象是真实世界的模拟就是这个道理。

箴言格语

相比较而言，面向对象程序设计的关键之处是匹配问题求解领域。结构化的程序设计方法提供了与基础硬件结构非常密切的一组抽象，而对象的抽象概念则提供了应用程序开发的基本背景，在这一点上，我们需要做更多的工作才能生成一个面向对象的应用程序。

类是面向对象技术的进一步深化。支持类的目的是提供一种基本的分类形式。类通过创建对象实例来支持这些对象共享完全相同的行为，它是创建对象的模板，包含了创建对象的

属性描述和行为定义。

总之，没有面向对象的基本特征就没有可视化的编程方法。

1.3.1 何谓对象

面向对象程序设计（Object-Oriented Programming，简记为 OOP）是 Kylix/Delphi 诞生的基础。OOP 立意于创建软件重用代码，具备更好地模拟现实世界的能力，这使得它被公认为自上而下编程的杰出典范。通过在 Pascal 语言中加入扩展语句，把函数“封装”进 X Window 编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰、编写容易。而 Kylix 是完全面向对象的，这就使得 Kylix 成为一种炙手可热的促进软件重用的开发工具，因而也就具有强大的吸引力。

箴言碎语

说面向对象是程序设计中的一场革命，不是对对象本身而言，而是对它们处理工作的能力而言的。对象并不与传统程序设计和编程方法相兼容，部分面向对象的方式往往会使情况更糟。除非整个开发环境都是面向对象的，否则对象产生的好处还没有它带来的麻烦多。

一些早期的具有 OOP 性能的程序开发工具如 KDevelop，虽然具有面向对象的特征，但不能轻松地刻画可视化对象，与用户交互能力也较差，程序员仍然要编写大量的代码。Kylix 的推出大大改变了这种现状。我们不必自己建立对象，只要在提供的程序框架中加入完成功能的代码，其余都交给 Kylix 去做。欲生成漂亮的界面和结构良好的程序再也不必绞尽脑汁，Kylix 将帮助我们轻松完成。Kylix 允许在一个具有真正 OOP 扩展的可视化编程环境中，使用它的面向对象语言。这种革命性的组合，使得可视化编程与面向对象的开发框架紧密地结合在一起。

1.3.2 X Window 中的对象

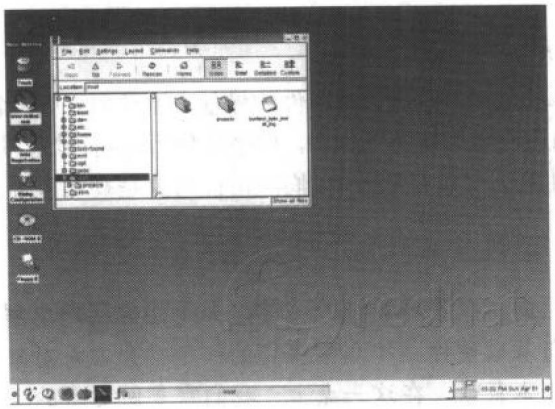


图 1.1 X Window 桌面系统包含的对象

上面的介绍也许会令部分读者感到晦涩难懂，不过在使用 X Window 的过程中，读者可一直在使用对象。在桌面上出现的任何一个图标都是对象，如图 1.1 所示。请注意当用户双