Kylix 编程入门

李巧云　宋航　程显峰　编著

机械工业出版社
CHINA MACHINE PRESS

Kylix 编程入门

李巧云　宋　航　程显峰　编著

机械工业出版社

本书以第一个运行在Linux操作系统下的、面向对象的、可视化的快速软件开发环境Kylix为学习研究对象，通过分析具体的程序实例从专业角度探讨了在Linux下开发应用程序遇到的问题和难点。本书教给读者如何能在最短的时间内掌握Kylix编程的基本概念，如果以前对Linux一无所知，也能通过本书前四章的简单介绍而熟悉和掌握Linux基本原理。

本书分为两大部分共9章，第一部分为从Delphi到Kylix，包括环境比较、初识Linux、使用库和Kylix程序员的C基础。第二部分为基本Kylix编程方法，包括进程控制、进程通信处理、文件系统的使用、在线帮助系统和系统管理。

本书适合具有一定编程基础的Delphi程序员学习在Linux下的编程开发。可供计算机程序开发人员、大专院校计算机专业师生、计算机网络爱好者和各种培训班学员参考使用。

图书在版编目（CIP）数据

Kylix编程入门/李巧云等编著. —北京：机械工业出版社，2003.4
ISBN 7-111-12058-2

Ⅰ.K… Ⅱ.李… Ⅲ.Linux操作系统—软件工具，

Kylix—程序设计 Ⅳ.TP.311.56

中国版本图书馆CIP数据核字（2003）第030260号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策　　划：边　萌　责任编辑：边　萌
封面设计：饶　薇　责任印制：闫　焱
北京京丰印刷厂印刷·新华书店北京发行所发行
2003年5月第1版第1次印刷
787mm×1092mm 1/16·17.75印张·438千字
0 001—4 000册
定价：33.00元（含1CD）

前　言

1991 年，当 Linus Torvalds 第一次对外公开发表 Linux 时，他自己可能也没有想到要把这项工作无限发展和延伸，开始只是作为一个娱乐项目，旨在使有兴趣的黑客都能熟练掌握 UNIX 操作系统，但是，Linux 很快迎合了许多业余程序员的口味。今天，世界上成千上万的程序员都在认真而积极地设法维护内核，而内核是 Linux 操作系统创建过程代码的关键部分。

目前，Linux 虽然还不够优越，但其高效及可靠性已被大多数用户和厂家认可。对于 Linux 而言，主要缺点是缺乏像 Windows 系统那样具有庞大的可利用的资源，然而，制约这个瓶颈的问题只能有两个：一是没有足够的时间和成本开发足够的 Linux 应用软件；二是缺乏开发上述应用程序的工具。然而，Kylix（即运行在 Linux 下的 Delphi）的发布已经开始改变这个状态了。Delphi 快速、高效的可视化开发环境与快速、稳定的低成本操作平台 Linux 之间已经完全融合。

Kylix是一个运行在Linux操作系统下的、面向对象的、可视化的快速软件开发环境。使用Kylix可以建立高效的基于Intel架构的32位Linux应用程序，并且具有高效紧凑的编程代码，Kylix提供了一整套开发工具，包括开发、测试、调试、发布应用程序等，Kylix还包含大量可以再利用的组件、一套设计工具、应用程序模板、编程开发精灵等。上述工具可以大大简化开发原型，缩短开发周期。

最新发布的Kylix 3版本包含了Kylix™ 3 Enterprise、Kylix™ 3 Professional和Kylix™ 3 Open Edition三个不同版本，Kylix 3真正成为一个跨平台的开发环境，不但同时包含了Delphi和C/C++编译器，并且可以共享Borland C++Builder和Delphi的Windows下的CLX（组件库），基于CLX的Kylix 3应用程序和源代码可以在C++Builder和Delphi中重新编译。Kylix 3广泛支持Oracle、IBM DB2、Informix、Red Hat Database、PostgreSQL、MySQL和InterBase数据库。该版本支持Red Hat 7.2、Mandrake™ 8.2和SuSE® Linux 7.3。

本书主要面向具有一定编程经验、但刚刚接触 Kylix 的 Delphi 程序员，通过许多实例分析，使读者慢慢熟悉这个新的操作系统下的开发工具。其次，本书教给读者如何能在最短的时间内运行 Kylix，如果以前对 Linux 一无所知也没有问题。本书的前四章对 Linux 的基本原理及其与 Windows 的区别作了一些简单介绍，并详细解释了 Linux 系统内 API 所用的 C 语言规则。对于具有一定编程经验的人来说，这是学习 Kylix 所必需了解的知识。

为了使读者能够在最短的时间找到解决编程问题的答案，我们对每一个问题的解析都不仅提供答案，而且详细阐明发现和解决问题的关键步骤。同时也涉及了在 Windows 下怎样发现该问题，以及它与解决 Linux 下的问题有何区别，解决问题的程序策略等，这对有一定编程经验的读者来说，能够起到举一反三，触类旁通的功效。

由于时间仓促，水平所限，书中错误在所难免，请读者批评指正。

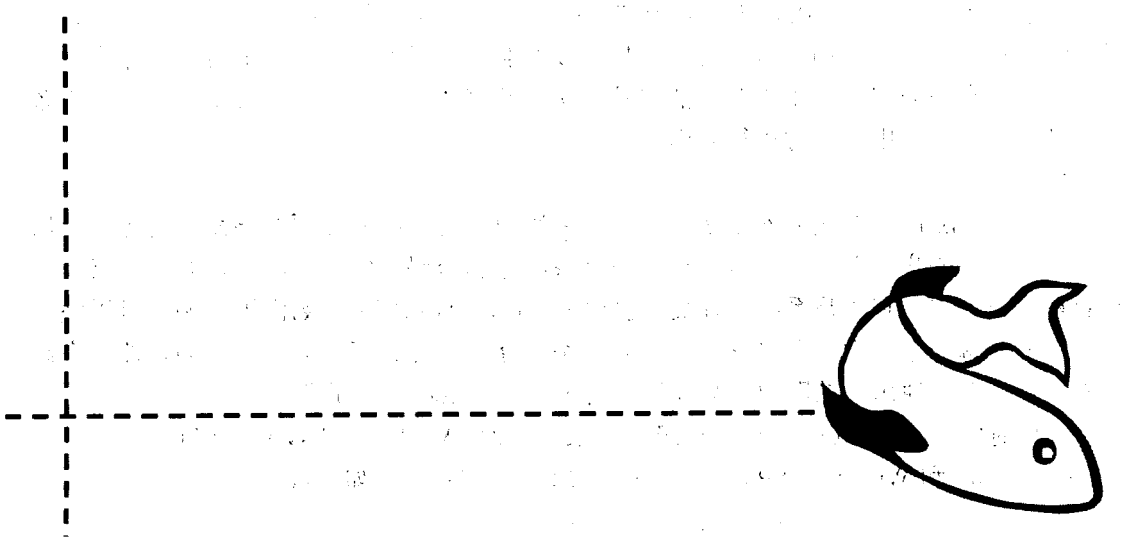
编　者

目　录

第二部分 基本 Kylix 编程方法

第一部分

从 Delphi 到 Kylix



第1章 环境比较

如果读者曾经读过一本关于 Linux 的书，就会知道 Linux 平台的可扩展性，并且其中不乏有令人难以攀登的一个个“山峰”。这时也许你想知道为什么 Linux 有些地方看起来与 MS-DOS 那么相似，比如那些黑白字符的界面和命令行，而 MS Windows 却与之截然不同。

如果处在一个陌生的环境中，想从一个地方到另一个地方，最好的办法就是拥有一张地图。本章是构成一张地图的四章内容中的第 1 章。

在本章中，我们将比较 Linux 和 Windows 这两种环境。尽管这种比较不能详细包容这两种环境之间的所有差异，但它至少可以提示 Windows 高手们在学习 Linux 时可能会走的弯路。对于 UNIX 或 Linux 开发高手来说，则可略过本章。

1.1 两种环境之间的差异

虽然 Linux 和 Windows 有许多的不同，但在很多方面惊人地相似。Windows 采用了许多 UNIX 的特征，并在某些情况下对它们进行了扩展和延伸，这就意味着它们有许多概念彼此相似，因此，可能走的弯路也不会太多。但是，通过提供一个快速的“概况浏览”，或许对学习这些东西有用。

1.1.1 系统的差异

Linux 和 Windows 之间最大的差异就在于，Linux 从开始就是作为一个真正的多用户系统设计的。这里的多用户（Multiuser）既不是指同时从一个网络驱动器访问数据的用户数量，也不是指在此系统拥有账户的用户数，而是指多个共存的、执行应用程序的用户，每一个用户都与坐在系统控制台前的用户没有什么区别。对于许多应用程序来说，这对编程算法及创建它们时所用的编程技术没有影响。然而，对于其他应用程序来说，同时访问共享资源将会产生许多新的问题。访问共享资源的程序（配置文件、硬件设备等）必须允许几个用户同时执行它们。

- 安全与许可

由于 Linux 的一个重要作用是作为一个多用户系统，因此需要给该系统的每一个用户都开设一个账户，并且赋予每一个账户登录账号和密码。每一个账户都有一个级别，有的是普通用户，有的是系统管理员（也称为 root，有时也指超级用户）。root 可以完全控制系统的每一个方面，并且它对系统有生杀予夺的权力。正因为这个原因，即使高级用户也有一个普通用户的账户，只在进行管理时才以 root 的形式登录。

编程时需牢记 Linux 具有用户级别的限制，而 Windows 则没有（Windows NT、Windows 2000 和 Windows XP 除外）。在多数 Linux 系统中，如果没有获得许可，普通用户不能装载软盘或 CD，不能装载网络驱动器，不能在他们自己主目录以外进行文件写操作（有些情况下读操作也被禁止），甚至也不能改变主机的系统设置，所有这些操作都

只能由 root 来完成。程序员（尤其是系统公用程序开发人员）必须知道，当一个非 root 用户运行时，他们想要其程序完成的工作并非都能完成。

提示：

尽管对于默认配置来说，普通用户不能配有软盘，但也有例外，如 Linux 组件 mtools 就能使普通用户访问和操作存储在 MS-DOS 格式化盘中的数据，这在绝大多数 Linux 装置内部都以默认方式存在，且支持绝大多数标准的 DOS 命令，但它们都有前缀 m（如 mcopy 和 mdel），详情见 mtools 操作说明。当然，系统管理员也可以通过修改 file/etc/fstab（当他们作为 root 登录时）而使普通用户可以装载文件系统，详情见 fstab 操作说明。

- 配置

如果经常使用 Windows 注册表，我们会非常惊讶地发现它完全是 Linux 所没有的东西。尽管有些开发者认为这是好事，但许多人则越来越“沉迷”于 Registry 了，这里用“沉迷”可能有点夸张。那些以前存放在系统配置数据库中的配置信息现在都将被存放在配置文件中，从某些方面来说，这与 Windows 3.x 出现时的情况惊人地相似（由 INI 文件所带来的复杂性的增加让人头痛）。然而，Linux 不存在 Windows 所具有的“单点失败”问题，在这种问题中，系统所有的配置信息都被存放在一个高度复杂的数据库中，这正是问题的症结。而 Linux 下的配置信息则存放在几个不同的位置，如表 1-1 所示。

表 1-1　配置信息的位置

信 息 类 型	位　　置
系统启动脚本	保存在/etc/rc.d 目录树下，这些文件每执行一次都进入一个新的运行层次
系统配置	配置文件放在/etc 中（例如/etc/hosts 和/etc/fstab）
全局应用程序配置	配置文件放在/usr/local/etc 或/etc 中
用户定制配置	“点文件”保存在用户的根目录下

在这里，需要对表 1-1 中的最后一行作一下解释。在 Linux 中，点文件指的是以字符（.）开头的文档和指令。它们是隐藏文档，在正常的指令列表中无法显示。为使它们在一指令列表中可见，必须在文件列表命令 Ls 中使用选项-a。这使得点文件即使不一块儿隐藏也不太容易注意到。比如，我们可以在自己的主指令中寻找文件.Xdefaults 或查找指令.kde。巧妙地运用点文件和点指令，就可以快速准确地模仿系统配置数据库的分枝 HKEY_CURRENT_USER 的功能。然而，当想要对以这种方式储存的配置文档的标准 API 所在的范围之内进行访问时——就不那么幸运了。但 Kylix 支持 INI 格式文件，这是在该方向上迈出 s 的一大步。

- 进程通信与库

另一个特殊的领域是 Linux 程序员为进程通信而设置的选项或 IPC。在 Linux 下，应用程序或进程有多种交流方式，从简单的信号到复杂的数据共享技术。这些有的在 Windows 下也有，有的则没有。关于 IPC 方法的详细情况见第 2 章。

最后一个方面的介绍在本书中占了整整一章，即 Windows 下的动态连接库。例如

Linux 下没有 DLLS，但有一个更灵活更复杂的装置——共享库。关于 Linux 共享库的详细情况见第 3 章。

1.1.2 开发环境的差异

现实有时候显得太残忍了，不是吗？与 Windows 的开发者所用的开发环境相比，Linux 的开发环境还相当落后。虽然在 Windows 中完整的开发环境很普遍，但在 Linux 中还很罕见。有经验的 C 程序员可能认为开发 Linux 程序真正需要的是编辑函数 vi 及一个程序文件的描述文件。从本质上来说，那是对的——但那只是长期以来把 Linux 编程限定在命令行环境的观点。

尽管 X-Windows 系统在 Linux 中的使用使得基于图形用户接口的程序得以在 Linux 中被开发及执行，但由于缺乏一个标准的、强大的基于 GUI 的开发工具，还不能认为它是一个有竞争力的、绝对专业的开发环境，也不能认为它是基于 GUI 的应用程序。Kylix 有望成为 Linux 的标准工具。尽管如此，过去 10 多年所用的许多命令行今天仍然很有用。

直到目前，在 Linux 的发展领域内，Pascal 还不是很强大。虽然有些命令行 Pascal 编译程序可用，包括 GNU 编译程序，但这种语言严重缺乏程序库及具体的语言工具。实际上，有一个更常用的 Pascal 工具 ptoc，可以把 Pascal 源代码转化为 C，这本身是对 Linux 中 Pascal 使用状态的一个不好的评价。Kylix 显然会彻底改变这种情况，它将给 Linux 中 Pascal 的发展带来新的生机。下面是传统开发工具和其用法的简单解释。

- make

根据相关规则进行自动构建。多用于日常构建的群体开发。

- cvs

面向项目的源代码修改管理。

- adb/gdb

用于协调邮件发送的调试工具（起源于 Kylix 的标准）。

- rpm

包管理程序，作为一个 de facto 标准包安装程序和管理工具。其他工具也有这个作用，但是，rpm 在 Linux 安装过程中使用得最广泛。

- man

显示命令、文件和系统调用的菜单页。详细情况请使用“man man”查看。

- zip/unzip

旧的压缩和解压缩程序（现在源代码则是开放的）。程序 tar 和 gzip 具有相似的功能且是一个假冒的 Linux 标准。如果你的系统没有 zip 和 unzip，也可以用它们来代替。

- grep

用于发现含有特定字符串或规则表达式的文件。

- shell scripts

用于自动执行重复任务，比如构建和打包。

校正控制或变化管理系统，如 SourceSafe、ClearCase 及 CMS 的使用很容易适应。如果你是组中的一部分并且以前没有使用过校正控制系统，强烈推荐试用一下。这与安全网络的使用不同，在后一种情况下，无论更改正在使用代码副本的情况多么糟，总能

恢复以前未受到破坏的版本或在这之前的版本。

Linux 中有一个常用的版本控制工具 CVS（共存版本系统）。CVS 允许开发人员同时使用代码的某些部分，并且可以安全地进行改变，并用一代码知识库跟踪它们（当然，独立的开发者也可以使用这个工具）。CVS 使得获得一个完整修订版或者发行版的源代码包比较容易，特别是在试图发现和修正两个老版本中的 Bug 时，就显得得心应手了。它在项目级别上进行操作，使用整个路径树而不是文档（这同更加低级的版本控制包如 SCCS 或 RCS 一样），且它可以使不使用版本控制而产生的执行无效。

1.1.3 开放资源的差异

如果不把开放资源的问题作为 Linux 和其相应的操作系统（只有 Windows 一个）之间的一个重要差异而提出，那么，我们似乎忽略了一个问题。这里，尽量避免把 Linux 和开放资源列表比较，但事实是，Linux 确实有一些显著的优点。

在编程这方面很明显：绝大多数开放资源（Open Source，即源代码开放）软件都能被自由复制及使用，没有登记注册的麻烦，因此，也就没有费用或合法性危险。例如，Linux 内核的源代码本身就可下载使用，用户可以任意改变它以适应自己的想法，但有一点必须明白，即这种改变不能在客户或客户机站点上出现，除非你能控制那些站点。主要软件包（如 MySQL 数据库包）正在采用具有开放资源的功能，这种情况经常发生。几乎所有今天可利用的开放资源的资料都是用 C 及 C++编写的（至少直到现在，斜体仍然表示 Linux 程序）。但我们必须清楚的是，并非代码的可用性构建了“开放资源”，而是其开发者们共同的观点，也是代码使用方式限制的一个改变。关于开放资源软件一个很形象的描述就是“同讲话一样自由，而非同在喝醉中一样自由”，意思是说，不是开放资源代码本身自由，而是说他在其他项目中使用时可以不受限制。

下面是开放资源对于程序开发者的第二个好处：作者的可联系性。在大多数情况下，我们可以直接或通过网络与一个程序的作者交流，他或她通常会直接回答我们的问题或者向我们推荐一个能够提供答案的人。试着与你所喜欢的操作系统的作者们进行这样的交流吧！虽然可能会为支持开放资源产品而支付一些费用，但这样就可以经常地与开发者们进行直接交流，甚至还可能有机会以某种方式访问他们。

最后一个优点体现在市场方面：适合于小中型企业的 Linux 的开发是一个重要的转折点，许多企业的人们认识到了基于 Linux 的计算机的竞争优势，这些计算机的操作系统具有很高的可靠性且很开放，但是得不到迅速推广应用的惟一原因是：为桌面.Enter. Kylix 所编写的应用软件太少。

1.2 Linux 中的视窗功能

学过操作系统或网络系统软件的人可能会知道，它是典型的一层层构建起来的，每一层对其下面的层都起着一定的作用。这与视窗及 Linux 相同：视窗的作用分两层，一层给出了视窗的能力，另一层则提供图画式的、类似于 Windows 下的环境。

1.2.1 X Window 系统

在所有视窗事件的根部，都有一个 X，那不是一个几何语句。X 表示的是 X Window 系统，这是马萨诸塞技术学院开发的一个软件包，能为该系统提供核心的、基本的 GUI 服务。X 有三个主要组成部分：X 服务器、X 客户机以及 X 协议。这些部分的组织从概念上来讲很简单，如图 1-1 所示。

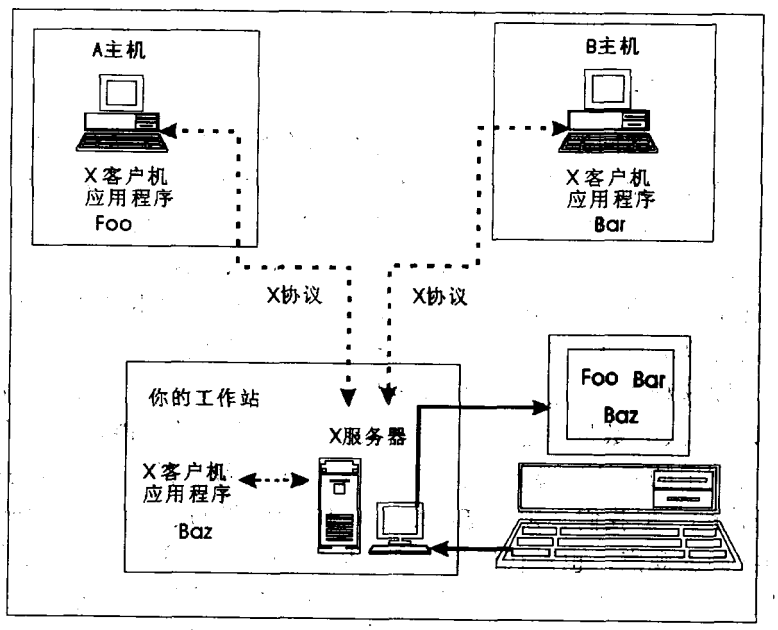


图 1-1 X Window 系统的主要组成

X 客户机由任何 X 兼容的应用程序来表示，如系统上所装备的配有 KDE 或 Gnome 的文本编辑函数。X 客户机是硬件中心，意思是说，它们没有专门显示或者输入代码的设备。X 客户机使用 X 协议交流屏幕信息，而输入设备则具有可在一个完全独立的机器上运行的 X 服务器。

X 服务器最终负责显示用户在应用程序的窗口上的所见，并且把输入内容重新分配给合适的 X 客户机。尽管 X 客户机是硬件中心，但 X 服务器是非常独立的设备。对于许多所谓“典型的”的图形适配器，一般都被 X 服务器发行版本的程序装载。在大多数 Linux 系统上，X 服务器都是 X Free86 包。其他情况下，X 服务器控制鼠标和键盘输入、屏幕分辨率、低级设备的基本要素及中心输出规则。它也提供一个非常基本的 GUI。XLib 库的作用就是开发该层次上的应用程序。然而，使用 Xlib 的应用程序需要绘制并建立其自己的屏幕元素（称为窗口小部件），并处理大量用户在 Windows 上已习惯的 GUI 工作。XLib 应用程序没有一般的“所见即所得”，并且通常很难开发和维护。要在 Windows 上绘制一条平行线，建立 XLib 应用程序与建立 Windows 3.0 下的 Windows 应用程序相似，只是更差。X Window 系统与 Windows 的主要区别在于，Windows 更独立。不可能把 Windows GUI 从 Windows 内核中分离出来，而在 Linux 中，GUI 和内核则是完全独立的

实体。

1.2.2 Window管理器

如果还没有深入到视窗系统的中心，用户可能会把绝大多数编程时间花费在窗口管理器的水平上。窗口管理器程序位于X和XLib的顶部，它是基本的X Window系统和用户之间的界面。它负责绘制和保持窗口环境的各个方面，如窗口边界、菜单、按钮键盘、工具栏及虚拟桌面，通常能够适应用户的需要和想法。

在Windows下，只有一个窗口管理器程序：Windows本身。而在Linux下，则可以从各种窗口管理器程序中选择所需要的。这次的竞争者主要是KDE和Gnome，其他还有诸如Window Maker、AfterStep及Enlightenment。但在本书中，我们将把重点放在两个主项（KDE和Gnome）上。其他的环境也有所贡献（比如支持声音），但目前它们正在运行一个遥远的第三方。

可用的窗口管理器程序确实很多，但要支持所有的窗口管理器程序并非那么容易，既需要Kylix的作用，又要靠Qt类库作者的想象力。

Borland做了一项优秀的工作，它创建了用Delphi装载的VCL类库，对艰难而复杂的Windows SDK的抽象简直是一个奇迹。它把这些复杂而细致的工作完成得很好，例如，显示屏幕元素、绘制图表、处理事件及进入GUI程序等所有繁琐的细节。这种抽象在Kylix中的新CLX框架中也有。无论窗口管理器程序显示与否，在Kylix存在的情况下，对象TButton都是GUI中的一个开关，意思是说，无论应用程序的用户是否正在运行KDE、Gnome或Enlightenment，其作用都将仍然如当初设计的一样。

然而，GUI程序的抽象都只能走到这一步。只抽象一个GUI不能使基本类透过视窗环境而工作，更不用说透过操作系统——Qt库的发源地了。当把VCL库运到CLX框架时，Borland用CLX把Qt包了一圈，然后在不同的环境下隐藏编程的细节。在Windows的Delphi存在的情况下，用户可以任意贬低窗口管理器程序（或者OS本身），但是在真正传统的Delphi中，由于Kylix的存在，这就没有必要了。

可能绝大多数读者不需要熟悉比用Kylix装载的CLX框架更深的内容，但即使具有X或Qt程序的基本知识，最好也要坚持使用CLX。使用CLX框架会使源代码从Linux到Windows（以及Borland将来决定要支持的任何其他平台）的转换更方便。

1.2.3 窗口小部件与组件

想知道为什么在Delphi中所熟悉和喜欢的“组件”到Kylix中却改名为“窗口小部件”了吗？原因就在于Borland已用Qt代替了Windows指定的Delphi组件，Qt是由挪威的Trolltech开发的与Linux对象兼容的一个相同的东西。这些类执行一系列X环境下的GUI控制（例如键盘、按钮、滚动条及字段），且在那个环境中，GUI控制总是用窗口小部件来表示。Qt窗口小部件与Delphi VCL组件（可见的组分库）非常相似，只是名字不同。

Qt库是一个对象定位的GUI软件工具，用C++编写，这比视窗环境都方便。自1995年第一个商业版本出现以来，它已用在多个应用程序中，且其应用还在继续增长。

1.3 相关内容查询网址

开发 KDE 时所用到的程序文献，可查阅 www.developer.kde.org。如果要从 Trolltech 获得关于 Qt 类库及相关产品的更多信息，可查阅 www.trolltech.com，有关 Gnome 窗口管理器程序的详细资料，请查阅 www.gnome.org。最后，如果想知道更多关于开放资源功能的信息，请查阅 www.opensource.org。

第 2 章　初识 Linux

通过总体浏览两个系统的基本情况之后，我们可以更深入地了解 Linux 和 Windows 平台的一些差异了。在本章中，读者可以了解在 Linux 下文件和指令是怎样构成的，并从一个程序员的观点粗略地熟悉一下 Linux 操作系统的几个方面。最后，我们介绍一些涉及 / proc 文件系统的内容。

需要说明的是：Linux（及其前身 UNIX）是一种工业化的产品，现在已有许多关于它的书。本书只是想给读者起一个抛砖引玉的作用，并不是介绍完整的 Linux。但是，对本章及后面两章所提供材料有个基本的理解，有助于编写在 Linux 下运行良好的应用程序。

2.1　文件与目录

同 Windows 及其前面的版本一样，MS-DOS、Linux 都在磁盘中以文件的形式保存程序及数据，这样就必须在目录结构内组织文件。我们不打算深入涉及具体的磁盘微观级别的组织过程，而是把用户关于 Windows 文件系统的看法同 Linux 的活动方式联系起来。

2.1.1　文件及目录名

Linux 下的文件及目录名与 Windows 下的 UNC 命名规则很相似，只是在协议及所能接受的风格方面都有一些差异。

- **巨大的差异**

在 Windows 下，如果目录列表中含有文件 OneTrueWay.pas，则它既可以被当做 onetrueway.pas，也可以被当做 ONETRUEWAY.PAS 来引用，Windows 都能识别，并将选择一个恰当的文件。在 Linux 下，这种情况将不再有效，文件名是很重要的，任何试图引用名字不匹配文件的操作都将失败，这里主要因为 Linux 对大小写敏感。因此如果“用 Windows 的方式思考”，就会感到那样的失败确实很神秘，简直令人搔头。

- **合法文件名与目录名**

Linux 文件名的用法与 Windows 非常相似，但是，也有很多东西可能会混淆。首先是右手的小指所指的键盘位置。Windows 由于某种原因，把向后的斜杠字符（\）作为一个目录分隔符来使用，而 Linux（同 UNIX 及许多其他操作系统一样）则使用向前的斜杠字符（ / ）。同 Windows 一样，Linux 的单个分隔符是系统中最上层目录（也称为根目录）的名字，而所有其他目录则是其子目录。Windows 的根目录数同其驱动器数一样多，而 Linux 则只有一个根目录，不管其驱动器有多少。在其他驱动器、组件及与其他计算机共享的驱动器上的文件系统也被统计在安装点上的本地文件系统中，其中，安装点只

是为此目的而添加到文件系统中的目录。

Linux 中的文件及目录名可以任意设置，共有 256 个字符，不过实际使用中很少有人使用那么多的字符，关于这个有些超出本书的范围了。文件及目录名可以包括除目录分隔符“斜杠”（ / ）之外的任何字符，但最好不要使用短杠（-）作为一个文件名的首字符。

2.1.2 关于安装的其他信息

安装——这个术语可能起源于大型计算机的早期，那时候，内存不但短缺且很贵，硬盘驱动器的尺寸有一台自动洗衣机那么大。后来为了使硬盘驱动器的尺寸变小，便用一个“磁盘组件”（同汽车上的一个轮胎大小差不多）代之，把它机械地安装到驱动器上，并把它固定。

在 Linux 下，在额外硬盘驱动器（计算机的硬件组成部分或网络共享部分）甚至个人磁盘目录上对数据的访问可以通过使用 mount 命令而被“安装”到整个系统上。在 Windows 中，新的磁盘驱动器总是被“安装”在 My Computer 层次上，且被赋予一个驱动器名。而 Linux 则把共享资源（驱动器或目录）与所定义的一个安装点（根据惯例，一个空的目录）联系起来。

如果想让软盘驱动器（在 Linux 下为装置标定 / dev/fd0）上的文件系统（目录及文件）变成 Linux 文件系统的一部分，可以在 / user 目录相邻的下一级目录 / mnt 得到。为此，则可使用下面的命令：

```
mount/dev/fd0/usr/mnt/floppy
```

Linux 在启动过程中使用文件/etc/fstab（file system table 的缩写）中预先定义的参数配置文件系统，并在指定的安装点安装驱动器。当插入新的媒介时，必须先卸下具有可移动媒介（如一个软盘）的驱动器，然后再安装。mtool 包（在第 1 章中已提到，且安装时自动带有许多 Linux 字段）将自动做装卸工作。

提示：

尽管字符（-）在一个文件名中是非常合法的，但如果把它作为一个文件或目录名的首字符，可能会遇到麻烦。这是因为，Linux 通常用短杠字符表示命令行字符“选项”（与 Windows 中的斜杠相似）。如果要使用某一命令行工具（如 rm）并把它作为一个命令行参数来说明，那么，这个工具将会把文件名当作一个命令行选项，这就会带来灾难性的后果。如果必须使用文件的这种类型，可以使用其全部合法名称（例如 /home/fflintstone/-xyzzy），或者用 mv 重新命名它（例如，mv-- -xyzzy xyzzy）以使操作更容易。如果感到迷惑，本例中的字符“双短杠”（--）会告诉 mv 不会再有命令行选项，以防止这个公用程序把-xyzzy 当做一个命令行选项来解释。Linux 中的许多命令行公用程序都支持这种规则，这使其能够随时被记住。

2.1.3 驱动器名字

Windows 和 Linux 文件系统的另一个容易混淆的差异就是驱动器名字。在 Windows