

第一部分 Linux GUI 编程

框架及编程基础

第1章 Linux软件开发概述

1.1 关于Linux

Linux于1991年诞生于芬兰。大学生Linus Torvalds，由于没有足够的钱购买昂贵的商用操作系统，于是自己编写了一个小的操作系统内核，这就是Linux的前身。Linus Torvalds将操作系统的源代码在Internet上公布，受到了计算机爱好者的热烈欢迎。各种各样的计算机高手不断地为它添加新的特性，并不断地提高它的稳定性。1994年，Linux 1.0正式发布。现在，Linux已经成为一个功能强劲的32位的操作系统。

严格地说，Linux只是一个操作系统内核。比较正式的称呼是GNU操作系统，它使用Linux内核。GNU的意思是GNU's not Unix(GNU不是Unix)——一种诙谐的说法，意指GNU是一种类Unix的操作系统。GNU计划是由自由软件的创始人Stallman在20世纪80年代提出的一个庞大的项目，目的是提供一个免费的类Unix的操作系统以及在上面运行的应用程序。GNU项目在初期进展并不顺利，特别是操作系统内核方面。Linux适时而出，由于它出色的性能，使它成为GNU项目的操作系统的内核。从此以后，GNU项目进展非常迅速：全世界的计算机高手已经为它贡献了非常多的应用程序和源代码。

Linux是遵从GPL协议的软件，也就是说，只要遵从GPL协议，就可以免费得到它的软件和源代码，并对它进行自由地修改。然而，对一般用户来说，从Internet或者其他途径获得这些源代码，然后对它们进行编译和安装是技术难度很高的工作。一些应用程序的安装也都非常复杂。因而，有一些公司如Red Hat、VA等开始介入Linux的业务。它们将Linux操作系统以及一些重要的应用程序打包，并提供较方便的安装界面。同时，还提供一些有偿的商业服务如技术支持等。这些公司所提供的产品一般称为Linux的发布版本。目前比较著名的Linux发布版本有以下几种：

Red Hat——最著名的Linux服务提供商，Intel、Dell等大公司都对其有较大投资，该公司前不久收购了开放源代码工具供应商Cygnus公司。

SlackWare——历史比较悠久，有一定的用户基础。

SUSE——在欧洲知名度较大。

TurboLinux——在亚洲，特别是日本用户较多。该公司在中国推出了TurboLinux 4.0、4.02和6.0的中文版，汉化做得很出色。

Debain——完全由计算机爱好者和Linux社区的计算机高手维护的Linux发布版本。

Linux进入中国后，在我国计算机界引起了强烈的反响，最近两年，也出现了许多汉化的Linux发布版本，影响较大的有以下几种：

XteamLinux——北京冲浪平台公司推出的产品，中国第一套汉化的Linux发布版本。

BluePoint——1999年底正式推出的产品，内核汉化技术颇受瞩目。

红旗Linux——中国科学院软件研究所和北大方正推出的 Linux 发布版本。

从本质上来说，上面所有发布版本使用的都是同样的内核(或者版本略有不同)，因而，它们在使用上基本上没有什么区别。但它们的安装界面不一样，所包含的应用程序也有所不同。

Linux之所以大受欢迎，不仅仅因为它是免费的，而且还有以下原因：

1) Linux是一个真正的抢占式多任务、多线程、多用户的操作系统。

2) Linux性能非常稳定，功能强劲，可以与最新的商用操作系统媲美。

3) Linux有非常广泛的平台适应性。它在基于 Intel公司的x86(也包括AMD、Cyrix、IDT)的计算机、基于Alpha的计算机，以及苹果、Sun、SGI等公司的计算机上都有相应的发布版本，甚至在AS/400这样的机器上都能找到相应的版本。Linux还可以在许多PDA和掌上电脑以及嵌入式设备上运行。

4) 已有非常多的应用程序可以在Linux上运行，大多数为SCO Unix开发的应用程序都能在Linux上运行(借助于iBCS软件包)，甚至还比在SCO Unix上运行速度更快。借助Dosemu，可以运行许多DOS应用程序，而借助Wabi或Wine，还可以运行许多为Windows设计的软件。

5) Linux是公开源代码的，也就是说，不用担心某公司会在系统中留下后门(软件开发商或程序员预留的，可以绕开正常安全机制进入系统的入口)。

6) 只要遵从GPL协议，就可以自由地对Linux进行修改和剪裁。

当然，Linux的优点决不止于此。对计算机专业人员来说，Linux及其相关应用程序也是学习编程的绝好材料，因为这些软件都提供了完整的源代码。

Linux的出现为我国软件产业赶超世界先进水平提供了极好的机遇，也为我国软件产业反对微软的垄断提供了有力的武器。

1.2 关于Linux的桌面环境

目前使用Linux主要在于服务器端。在Internet上有很多服务器都在使用Linux。但是，一个操作系统要想得到普及，并占据一定的市场份额，必须要使非计算机专业人士都可以轻松掌握这种系统。而Linux作为一种类Unix操作系统，对它的操作一般都是通过复杂的Shell命令进行的。因而，应该有一种简便易学的图形用户接口(Graphics User Interface, GUI)，使用户使用鼠标就可以完成大多数工作。

在Linux中，GUI由以下几个部分组成：

- 窗口系统—组织显示屏上的图形输出并执行基本的文本和绘图功能。
- 窗口管理器—负责对窗口的操作(比如最小化、最大化、关闭按钮的形状，窗口边框外观等)以及输入焦点的管理。
- 工具包—带有明确定义的编程界面的常规库。
- 风格—指定应用程序的用户界面外观和行为。

在Linux发展的初期，众多的计算机专家为它贡献了多种图形用户接口，如FVWM95、AfterStep等。这些接口模仿了Windows 95、Macintosh、NestStep、Amiga、Unix CDE等桌面环境。这些GUI在一定程度上来说只是其他图形接口的仿制品，不能提供优秀的操作系统所需要的特性。其后，自由软件社区的一批计算机专家开始了KDE项目(K Desktop Environment, K桌面环境)，目的是提供一个开放源代码的图形用户接口和开发环境。该项目取得了极大的

成功，KDE成为许多Linux发布版本的首选桌面环境。GNU/Linux项目因此而得到蓬勃发展。但是，KDE是基于Troll Technologies公司的Qt库的。Qt库是一个跨平台的C++类库，可以用于多种Unix、Linux、Win32等操作系统。Qt并不是遵从GPL或LGPL协议的软件包。它的许可条件是：如果使用它的免费版本开发应用程序或程序库，则所开发的软件必须开放源代码；如果使用它的商用版本，则可以用以开发私有的商用软件。另外，Qt库是属于Troll公司的产品，一旦Troll公司破产，或者被收购，自由软件事业将受到严重打击。

1997年由墨西哥国立自治大学的Miguel de Icaza领导的项目组开始了Gnome开发计划。Gnome是GNU Network Object Model Environment(GNU，网络对象模型环境)的缩写。该计划的最初目的是创建一种基于应用程序对象的架构，类似于微软公司的OLE和COM技术。然而，随着项目的进展，项目的范围也迅速地扩大；项目开发过程中有数百名程序员加入进来，编写了成千上万行的源代码。该项目进展很快，1998年发布了Gnome 1.0。目前的最新版本是于1999年10月发布的October Gnome。现在，Gnome已成为一个强劲的GUI应用程序开发框架，并且可以在任何一种Unix系统下运行。Gnome使用的图形库是Gtk+——最初为了编写GIMP而创建的一套构件库，它是基于LGPL创建的，可以用它来开发开放源代码的自由软件，也可以开发不开放源代码的商用软件。Gnome的界面与KDE的界面是类似的(Gnome的目的之一就是创建一套类似KDE的桌面环境)，熟悉KDE的用户无需学习就能够使用Gnome。由于以上几个原因，Gnome已经成为大多数Linux发布版本的首选桌面环境。

由于Gnome项目的成功，1998年11月Qt库的开发者Troll公司宣布修改许可证协议，Qt库将成为自由软件。但是获取Qt库的许可证很不方便，况且Gnome的进展也很不错，因而，只要有可能，应该避免使用Qt库以及KDE。

从用户的角度看，Gnome是一个集成桌面环境和应用程序的套件。从程序员的角度看，它是一个应用程序开发框架(由数目众多的实用函数库组成)。即使用户不运行Gnome桌面环境，用Gnome编写的应用程序也可以正常运行，但是这些应用程序是可以很好地和Gnome桌面环境集成的。Gnome桌面环境包含文件管理器，它用于任务切换、启动程序以及放置其他程序的“面板”、“控制中心”(包括配置系统的程序以及一些小东西)等。这些程序在易用的图形界面背后隐藏了传统的UNIX Shell。Gnome的开发结构使开发一致的、易用的和可互相操作的应用程序成为可能。

1.3 Linux系统中的软件开发

1.3.1 开发所使用的库

在Linux下开发GUI程序的首要问题是采用什么样的图形库。在Linux的发展历史中曾经出现过多种图形库，但是由于自由软件的特点(没有技术方面的承诺)，使得无人继续对它们进行维护，或者其他方面的原因，这些库都已慢慢地被人遗忘了。

Gtk+(GIMP ToolKit，GIMP工具包)是一个用于创造图形用户接口的图形库。Gtk+是基于LGPL授权的，因此可以用Gtk+开发开放源码软件、自由软件，甚至商业的、非自由的软件，并且不需要为授权费或版权费花费一分钱。之所以被称为GIMP工具包因为它最初用于开发“通用图片处理程序”(General Image Manipulation Program，GIMP)，但是Gtk+已在大量软件项目，包括Gnome中得到了广泛应用。Gtk+是在Gdk(GIMP Drawing Kit，GIMP绘图包)的基

础上创建的。Gdk是对低级窗口函数的包装(对X window系统来说就是Xlib)。

读者可能会看到,在本书中既有GTK,又出现了Gtk+。一般用GTK代表软件包和共享库,用Gtk+代表GTK的图形构件集。

GTK的主要作者是:

```
Peter Mattis pe@xsf.berkeley.edu
Spencer Kimball spend@xsf.berkeley.edu
Josh MacDonald jma@xsf.berkeley.edu
```

Gtk+图形库使用一系列称为“构件”的对象来创建应用程序的图形用户接口。它提供了窗口、标签、命令按钮、开关按钮、检查按钮、无线按钮、框架、列表框、组合框、树、列表视图、笔记本、状态条等构件。可以用它们来构造非常丰富的用户界面。

在用Gtk+开发Gnome的过程中,由于实际需要,在上面的构件基础上,又开发了一些新构件。一般把这些构件称为Gnome构件(与Gtk+构件相对应)。这些构件都是Gtk+构件库的补充,它们提供了许多Gtk+构件没有的功能。从本质上来说,Gtk+构件和Gnome构件是完全类似的东西。

GTK本质上是面向对象的应用程序编程接口(API)。虽然完全是用C写成的,但它仍然是用类和回调函数(指向函数的指针)的方法实现的。

1.3.2 Gnome的开发结构

只使用Gtk+构件也可以开发出优秀的Linux应用程序,但是Gnome构件,特别是GnomeApp、GnomeUIInfo等,使开发界面一致的应用程序变得更加容易。Gnome的一些新特性,如popt参数分析,保存应用程序设置等也是Gtk+构件所没有的。

Gnome的应用程序开发结构核心是一套库,都是由通用的ANSI C语言编写的,并且倾向于使用在类UNIX的系统上。其中涉及图形的库依赖于XWindow系统。Gnome差不多对任何语言都提供了Gnome API接口,其中包括Ada、Scheme、Python、Perl、Tom、Eiffel、Dylan以及Objective C等。至少有三种不同的C++封装。本书只介绍有关库的C语言接口,不过,对使用其他语言绑定的用户来说,它也很有用,因为从C到其他语言之间的转换都是非常直接的。本书包含Gnome库1.0版本(包括兼容的bug补丁版,比如1.0.9——所有1.0.x版本都是兼容的)。

Gnome的开发架构包含以下一些内容:

1. 非Gnome 库

Gnome并不是从头开始的,它充分继承了自由软件的传统——其中许多内容来自于Gnome项目开始之前的一些函数库。其中一些库Gnome应用程序开发架构的一部分,但是不属于Gnome库——我们称之为非Gnome库。可以在Gnome环境中使用这些库函数。主要有以下几种:

Glib Glib是Gnome的基础,它是一个C工具库,提供了创建和操作常用数据结构的实用函数。它也涉及到了可移植性问题,例如,许多系统缺乏snprintf()函数,但是glib包含了一个,称为g_snprintf(),它能保证在所有平台上使用,并且比snprintf()更安全(它总是将目标字符串以NULL结尾)。Gnome 1.0中使用glib的1.2版本,可以和任何1.2系列的glib一起工作(1.2.1、1.2.2,等等)。

Gtk+ Gtk+(GIMP Toolkit的缩写),是在Gnome应用程序中使用的GUI工具包。Gtk+最初是为了设计GIMP而引入的(GNU 图片处理程序),但是现在已变成通用的库。Gtk+依赖于glib。Gtk+包中包含了Gdk,它是对底层的 X Window系统库Xlib的简化。由于Gtk+使用了Gdk而不是直接调用Xlib,因此Gdk的移植版本允许Gtk+运行在不同于X 但只有相对较少的修改的窗口系统上。Gtk+和Gimp已经移植到了Win32平台(32位的Windows平台,包括 Windows 95/98、Windows NT/2000)上。

对Gnome应用程序来说, Gtk+具有以下特性:

- 1) 动态类型系统。
- 2) 用C语言编写的对象系统,可实现继承、类型检验,以及信号 /回调函数的基础结构。
- 3) 类型和对象系统不是特别针对 GUI的。
- 4) GtkWidget对象使用对象系统,它定义了Gtk+的图形组件的使用接口。
- 5) 大量的GtkWidget子类(构件)。

Gnome在基本Gtk+构件集合的基础上添加了许多其他构件。 Gnome 1.0是在Gtk+ 1.2版本的基础上完成的。

ORBit ORBit是一个用C开发的CORBA 2.2 ORB。和其他ORB相比,它短小精悍,但速度更快,同时还支持 C语言映射。ORBit是以一整套库函数的方式实现的。CORBA,或称作通用对象请求中介构架(Common Object Request Broker Architecture),是一套对象请求中介,或称为ORB的规范。一个ORB更类似于动态链接程序,但是它以对象的方式操作,而非子程序调用。在执行过程中,程序能够请求一个特定的对象服务; ORB可定位对象并且创建对象和程序连接。例如,一个电子邮件程序可以请求 addressbook对象,并且利用它查找人名。与动态链接库不同, CORBA可以在网络内很好地运行,并且允许不同编程语言和操作系统之间进行交互。如果熟悉Windows操作系统下的DCOM,那么CORBA与之类似。

Imlib Imlib(图片库)提供一些例程,其中包括加载、存储、显示,以及定绘制各种流行的图像格式(包括 GIF、JPEG、PNG以及TIFF)的函数。它包括两种版本: Xlib-only版本和基于Gdk的版本。Gnome使用 Gdk版本。

2. Gnome库

下面所介绍的库是Gnome-libs包的一部分,并且是专门为Gnome项目开发的。

libgnome libgnome是一些与图形用户接口无关的函数集合, Gnome应用程序可以调用其中的函数。它包含分析配置文件的代码,也包含与一些外部实用程序的接口,比如国际化编程接口(通过GNU gettext包)、变量解析(通过popt包)、声音编程接口(通过Enlightenment Daemon, esound)等。Gnome-libs包考虑了与外部库之间的交互,因此程序员无需关心库的实现或可用性。

libgnomeui libgnomeui包含了与GUI相关的Gnome代码。它由为增强和扩展Gtk+功能而设计的构件组成。Gnome构件通常使用用户接口策略,以提供更方便的 API函数(这样程序员需要指定的东西较少)。当然,这也让应用程序界面更一致。

libgnomeui主要包含:

1) GnomeApp构件 一般用来为应用程序创建主窗口。它使用 GnomeDock构件,允许用户重新排列工具栏,还可以将工具条从窗口上拖开。

2) GnomeCanvas构件 用来编写复杂的、无闪烁的定制构件。

3) Gnome 内置的 pixmap(包括打开、关闭、保存以及其他操作的图标) 用于创建和使用对话框的例程。GnomePixmap 构件比 GtkPixmap 功能更多。

libgnomeui 中还有几种其他构件, 如 GnomeEntry、GnomeFilePicker 等。这些构件都比 Gtk+ 构件库中的构件功能更强, 也更方便。

libgnorba libgnorba 提供与 CORBA 相关的实用程序, 包括安全机制和对象激活。对象激活是指获得实现给定接口对象的引用过程, 它包括执行服务器程序, 加载共享库模块, 或为已有程序请求新的对象实例等。

libzvt 这个库包含一个终端构件(ZvtTerm), 可以在 Gnome 程序中使用它。

libart_lgpl 这个库包含由 Raph Levien 编写的图形绘制例程。在这里包含的是在 LGPL 许可下发布的, 用在 GnomeCanvas 构件中的, Raph Levien 也销售它的增强版本。实质上它是一个矢量图形光栅图形库, 功能类似于 PostScript 语言。

3. 其他库

这些库一般使用在 Gnome 应用程序中, 但它不是 Gnome-libs 专属的部分。

Gnome-print Gnome-print 目前还是实验性的, 但是非常有前途。它使用 libart_lgpl 库, 可以和 GnomeCanvas 一起工作得很好。它提供一个虚拟输出设备(称“打印上下文”), 因此一段代码能输出到一个打印预览构件或 PostScript 文件, 还可以输出到其他打印机格式。Gnome-print 也包含与打印相关的 GUI 元素, 例如打印设置对话框、虚拟字体接口(处理 X 字体不可打印的问题)。

Gnome-xml Gnome-xml 是还未经验证的 XML 引擎, 它由 WWW 协会的 Daniel Veillard 编写。它能按照树状结构分析 XML, 也能按照 XML 输出树状结构。它对任何需要加载和保存结构化数据的应用程序来说是很很有用的, 许多 Gnome 应用程序把它作为文件格式使用。这个库不依赖于任何其他的库(甚至 glib), 所以它只是在名义上是一个 Gnome 库。然而, 可以认为大多数 Gnome 用户都安装了它, 因此如果应用程序使用了这个库, 对用户来说也没有什么不方便。

Guile Guile 是 Scheme 编程语言在一个库中的实现, 它使任何应用程序都能带有一个嵌入式的 Scheme 解释器。它是 GNU 项目的正式扩展语言, 并且有一些 Gnome 应用程序也使用它。为应用程序添加扩展语言听起来挺复杂, 但是有了 Guile 后就微不足道了。一些 Gnome 应用程序也支持 Perl 和 Python, 一旦实现了应用程序, 同时支持几种语言就会变得很容易。Guile 在 Gnome 开发者心目中有着特殊的地位。

Bonobo Bonobo 是一种对象嵌入式结构, 类似于 Microsoft 的 OLE。例如, 它允许你在电子表格中嵌入图表。它将在 Gnome 中普遍使用。任何应用程序将能通过适当的 Bonobo 组件调用 Gnome 库, 显示 MIME 类型数据, 例如纯文本、HTML 或图像。

1.4 开发 Linux 应用程序的编程语言和编程工具

Linux 是一种类 Unix 的操作系统。传统 Unix 下的开发语言是 C 语言。因为 C 语言是平台适用性最强的语言, 差不多每种平台上都会有一个 C 编译器。C 语言也更易移植, 因而, 在 Linux 下编程的最佳语言应该是 C 语言, Linux 上的很多应用程序就是用 C 语言写的。当然, 也可以使用其他语言。

因为 Gtk+ 和 Gnome 是用 C 语言编写的, 所以在开发 Linux 下的 GUI 程序时使用 C 语言是非常

方便的。但是 Gtk+ 也提供与许多其他语言的接口，如 Ada、Scheme、Python、Perl、Tom、Eiffel、Dylan 以及 Objective C 等。如果用 C++ 语言开发基于 Gtk+ 应用程序，可以使用一个名为 Gtk-- 的函数库，它是 GTK 工具包的 C++ 风格的封装。如果要用 Gtk+ 库和其他语言，最好参考相应的文档。本书只介绍使用 C 语言开发 Linux 程序。

一般的 Linux 发布版本中都提供了 C 编译器 gcc 或 egcs。使用 gcc 或 egcs 可以编译 C 和 C++ 源代码，编译出的目标代码质量非常好，编译速度也很快。

各种 C 编译器都要使用一些 C 语言实用函数。为了保证程序的可移植性，gcc 没有使用通用的 C 函数库，而是使用一种称为 glib 的函数库。glib 也是 Gtk+ 的基础。它包含一些标准函数的替代函数（如字符串处理函数）和基本数据结构的实现（单向链表、双向链表、树、哈希表等）。glib 中所包含的函数消除了某些函数的安全漏洞，使其更加可靠，在不同平台上移植也更加方便。

还有许多使用工具可以提高 Linux 下的编程效率，如 gdb 是优秀的 C 语言调试器，有丰富的调试指令；automake 和 autoconf 用于由源代码结构配置编译选项，生成编译所需的 Makefile 文件。

到目前为止，还没有像 Windows 平台上的 Visual Basic、Delphi 等一样的可视化的快速应用程序开发工具。开发 Linux 应用需要用文本编辑器书写源代码，然后再用编译器生成应用程序。眼下有一个开发小组正致力于开发一个 Linux 下的类似于 Visual Basic 的开发工具——gBasic，另外，预计 Inprise 公司（即 Borland）的 Delphi for Linux 也即将面市。

有几种正在开发的 RAD (Rapid Application Development) 工具，其中最希望的是 Glade——一种 GUI 生成器，可以快速生成创建界面的 C 源程序。

1.5 本书的结构

本书包含了以下内容：

- Gnome 应用程序开发的框架。
- 开发 Linux 应用程序的方法和步骤。
- glib，Gtk+/Gnome 构件的使用方法。
- GDK 基础知识。
- Linux 常用编程工具：调试工具 gdb，GUI 生成器 Glade。

本书附录包含 Gtk+ 和 Gnome 对象介绍（每个对象有一个简短描述），还有一些在线编程资源。

本书提供了大量可供参考的源代码。如果觉得内容不够充分，请参考相应的头文件。实际上，Gtk+/Gnome 和 glib 的头文件都是非常简单易懂的，从函数名称就可以猜到它的用处和用法。如果还觉得不够，可以钻研它们的源代码，这对了解它们的实现方法也是极有好处的。