万水计算机编程技术与应用系列

UNIX/Linux 图形开发集粹

裴 植　肖 薇　等编著

中国水利水电出版社
www.waterpub.com.cn

万水计算机编程技术与应用系列

UNIX/Linux 图形开发集粹

裴植　肖薇　等编著

中国水利水电出版社

内 容 提 要

本书主要介绍了 UNIX/Linux 图形化开发的基础知识、实用开发思想及实际技巧。本书将互联网时代通用的 XML 语言引入 UNIX/Linux 图形化开发中，同时介绍 CORBA 开发思想的应用，并配以大量的示例程序和讲解，方便读者的学习。本书主要包括如下内容：GNOME 的体系结构及函数库知识；创建 GNOME 程序的源代码树及窗口程序的方法；XML 语言在图形程序中的应用；有关 GDK 的开发方法及技巧；ORBit/CORBA 在 UNIX/Linux 图形环境下的开发方法；GTK+的对象和类型系统的概念及开发知识；通过 Widget 工具进行图形化开发的详细介绍；图形环境开发实例详解。

本书面向广大 UNIX/Linux 编程人员以及图形化开发的爱好者，以其通俗易懂的表述方式，对于尚未具备图形化开发概念的初学者以及中级开发人员的学习都会大有帮助。

图书在版编目（CIP）数据

UNIX/Linux 图形开发集粹 / 裴植，肖薇等编著. —北京：中国水利水电出版社，2002

（万水计算机编程技术与应用系列）

ISBN 7-5084-1046-7

Ⅰ.U… Ⅱ.①裴…②肖… Ⅲ. ①UNIX 操作系统－软件开发②Linux 操作系统－软件开发 Ⅳ. TP316.81

中国版本图书馆 CIP 数据核字（2002）第 024973 号

书　　名	UNIX/Linux 图形开发集粹
作　　者	裴植　肖薇　等编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）68359286（万水）　63202266（总机）　68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1092 毫米　16 开本　19 印张　414 千字
版　　次	2002 年 5 月第一版　2002 年 5 月北京第一次印刷
印　　数	0001—4000 册
定　　价	28.00 元

前　言

众所周知，Internet 思想来自于 UNIX。学习 UNIX 是掌握 Internet 技术精髓的关键。该操作系统早已经被公认为是电子商务领域以及众多关键性业务的理想平台，拥有着广泛的市场基础。而 Linux 操作系统是后起之秀，凭借免费获取的优势在中小企业和个人用户中有了一席之地。Linux 操作系统的技术根基是 UNIX，二者有着很多相似之处。但是，由于它们都缺少容易使用的桌面系统，妨碍了在商业、家庭和办公领域的长远发展。近年来，在 UNIX 和 Linux 平台上的图形化开发技术日趋进步，XML 语言的深入人心以及 CORBA 中间件技术的广泛应用，促使图形化开发环境迅速进步，开发需求也与日俱增。因此，现在的 UNIX/Linux 开发人员急需一些符合 Internet 发展潮流的技术读物。本书因此应运而生。

GNOME 也是一种用于 UNIX 系统的功能强大，成熟的桌面环境。它为用户提供了一个包罗万象的核心函数库，开发人员只需要把经历集中于 GNOME 函数库上，就可以以最快的速度，最小的精力开发出 GNOME 兼容的、一致的、非常强大的 GUI 应用程序。GNOME 数据库中包括了创建图形用户界面的函数，创建统一外观的高层组件，一个快速而且简单的 CORBA ORB，处理配置文件的各种函数。GNOME 同时提供了处理 XML 数据和 HTTP 连接的函数库。更重要的是，GNOME 提供了很久依赖软件系统中所缺少的结构，如组件结构，打印和字体结构等等。

在有关 UNIX/Linux 的书籍大量涌现的今天，我们推出了这本视角独特而且更加实用的开发类书籍。它弥补了 UNIX/Linux 图书领域图形化开发书籍的空白。本书面向的是 UNIX 和 Linux 系统的初中级开发人员，帮助他们了解和掌握图形化开发的思想和方法，在最短的时间内，在图形化环境中开发出实用而高效的程序。本书的出版将推动 UNIX/Linux 图形环境下程序开发的普及和进步，使程序设计者通过自学就可以开发出更多适合家庭和办公领域用户使用的优秀的图形化程序。它为那些需要学习 UNIX/Linux 图形化应用开发的人员提供了丰富的资料。

本书的编写工作由肖薇、裴植、肖蓬、姚亚元、张建、潘敏、毕小东、方缘、何捷、穆宁、高红、张晓琴、某军、蔡平、刘南等完成，其中第一章至第三章由肖薇撰写，第四章至第六章由肖薇撰写，第七章至第九章由裴植撰写。由于筹备时间有限，书中难免出现纰漏，请读者多加指正。

编者

2002 年 3 月

目　　录

第一章　Gnome 开发基础

1.1　什么是 Gnome

Gnome 是 Mexican Autonomous National University 的 Miguel de Icaza 和全世界的一些 GNU 开发人员在 1997 年开始的一个免费软件开发项目。受类似的 KDE（K Desktop Environment）项目成功的激励，以及 GNU/Linux 操作系统的流行和 GTK+图形工具包的强大功能，Gnome 发展迅速——在一年之中，数以百计的开发人员加入到这个项目中来，编写了几千行的代码。Gnome 成为 GUI 应用程序开发的强大基础架构，它可以运行于多种不同的 UNIX 上。

“Gnome”实际是一个缩写：GNU Network Object Model Environment。这个项目本来是要创建一个为应用程序对象提供服务的基础结构，与 Microsoft 的 OLE 和 COM 技术类似。然而，项目的范围迅速扩大，在把名称中的“网络对象”部分变为现实之前，还需要做一些基础工作。Gnome 的最新版本中包括了一个对象嵌入结构，称为 Bonobo，Gnome 1.0 中包括了一个快速的、轻量级的 CORBA 2.2 ORB 称为 ORBit。

Gnome 是 GNU 项目的一部分，它全面的目标是开发一个免费的操作系统（命名为 GNU）以及它的应用程序。我们可以从http://www.gnu.org 中得到关于 GNU 更多的信息。

Gnome 有两个重要的方面。从用户角度看，它是一个集成的桌面环境以及应用程序集合，从开发人员的角度来看，它是一个应用程序的基础开发结构（由各种有用的函数库组成）。用 Gnome 函数库开发的程序在用户没有运行桌面环境的时候也可以正常运行，如果 Gnome 桌面存在的话，它们可以结合得非常好。

桌面环境中包括一个文件管理器，一个用于任务切换的“面板”，装载应用程序以及用于配置的“控制中心”和很多小的内容。这些程序把传统的 UNIX shell 隐藏在一个容易使用的图形界面之后。

Gnome 的开发基础结构允许编写统一的、容易使用的、内部可以操作的应用程序。X Window 系统设计者的设计思想是对开发人员不强加任何用户界面策略。Gnome 添加了一个“策略层”，它创建了一个统一的视觉和感觉环境。已经完成的 Gnome 应用程序可以与 Gnome 桌面环境协调工作，但是也可以单独使用——用户只需要安装 Gnome 的共享函数库。而且，我们还可以开发不依赖于 X Window 系统的 Gnome 应用程序。

可以简单地说，一个 Gnome 程序是一个 GTK+ GUI 的应用程序，它使用 Gnome 函数库。Gnome 函数库使得所有程序都有相似的外观和感觉，而且也简化了程序的开发过程。而且

Gnome 对 GTK+ 进行了扩展，加入了一组 GTK+中没有的 widget。

1.2 Gnome 开发体系的基础结构

Gnome 的开发基础结构是围绕一组函数库的，所有的内容都是用可移植的 ANSI C 语言写成的，可以用于所有类 UNIX 系统。与图形有关的函数库依赖于 X Window 系统。 而且有一些包装，把 Gnome API 导出到几乎任何语言中，包括 Ada, Scheme, Python, Perl, Tom, Eiffel, Dylan 和 Objective C。至少有三种不同的 C++ 包装。

我们将介绍 C 语言与函数库的接口；然而对于任何语言捆绑都是有用的，因为从 C 语言到自己所熟悉的语言的映射一般都是直接的。这里介绍了 Gnome 函数库的 1.0 版本。

在深入讨论 Gnome 函数库之前，我们先要看一下在一个 Gnome 应用程序之中不同的库所适用的部分。Gnome 函数库是最高层的。GTK+由两个部分组成，GTK 和 GDK。GTK 层为 C 语言提供了一个对象模型，并为 UI 工具包提供了最基本的 widget，它是上层 GUI 派生的基础。GTK 是依赖于 GDK 的，GDK 是 Xlib 的一个底层包装，Xlib 库直接与 X 服务器对话。除了 Xlib 以外的任何部分都是基于 GLib 的，它是一个非常有用的 C 函数库，提供了很多实用程序和可移植的功能以及一组容易被 C 语言使用的容器。

一个 Gnome 程序使用多个库组成的层次结构，如图 1-1 所示。Gnome 库在最高层，它包括帮助例程，类和特殊的 widget （例如， GnomeCanvas， 一个快速高层的绘画 widget），并为应用程序提供了一个基础架构。

该层次结构的第二层是 GTK，它是 GTK+库的一部分。这个库提供了基本的工具包和 widget（例如，按钮，标签和输入框）来创建 GUI 应用程序。大多数 GUI 是直接用 GTK 编写的。GTK 同时为 Gnome 库提供了一个功能强大的对象系统。

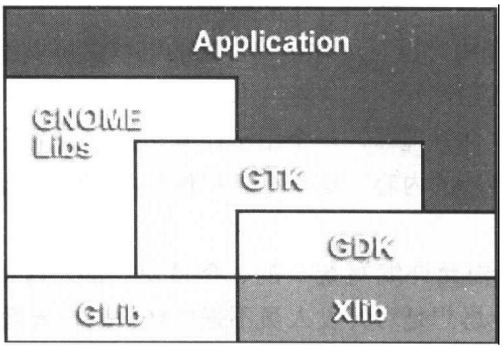

图 1-1 Gnome 应用程序的基本结构

再下面的一层是 GDK，它简单地对 X 函数库进行了包装，只有在我们进行特殊绘画或者设置窗口的特殊属性的时候，我们才需要用到它。

最下面的一层是 Glib，它是 C 的一个实用程序库，包括可移植性和一些实用程序功能，

以及一些容器类，例如，链表，可变数组，变长字符串，HASH，缓存，一个事件循环和其他有用的结构。

我们不要以为只有 C 语言才能开发 Gnome 应用程序。实际上，由于 GTK+和 Gnome 是用 C 语言编写的，它可以非常简便地和其他语言捆绑，例如 Objective C，C++，Perl，Guile，Python 和 Eiffel。

Gnome 库可以分为三个基本部分：libgnome，libgnomeui 和 libgnorba。libgnome 是一个实用程序函数库，它的设计思想与 Glib 类似，提供的服务包括：配置装载和保存，应用程序装载，MIME 类型标识和 MEDA 数据存储。libgnomeui 的部分，负责用户接口的工作，包括了应用程序基础架构，画布和多种其他特定的 widget。libgnorba 用来把 GTK+/Gnome 应用程序与 CORBA 集成在一起。

在使用 Gnome 开发过程中，一定要学会使用 GTK+。所有的 libgnomeui 类都是基于 GTK+的，它们的使用方法与标准 GTK+ widget 类似。

GTK+是一个基于容器的工具包，大多数 widget 都可以用做其他 widget 的容器。例如，一个按钮容器通常包括一个标签 widget。但是有几种 widget 的功能就是用于装载几个 widget，例如 horizontal box 和 vertical box 按行列的方式排列 widget，table 可以按照一定格式排列 widget。

这种层次结构允许 widget 在用户不计算的情况下，修改几何大小。就是说应用程序可以在不同的视觉效果下工作（例如，修改字体大小），并且在应用程序调整窗口大小的时候进行对应的调整。在运行时刻，它也非常容易用于开发自动生成的 GUI。图 1-2 所示的是 GTK+应用程序中的 widget 层次关系。

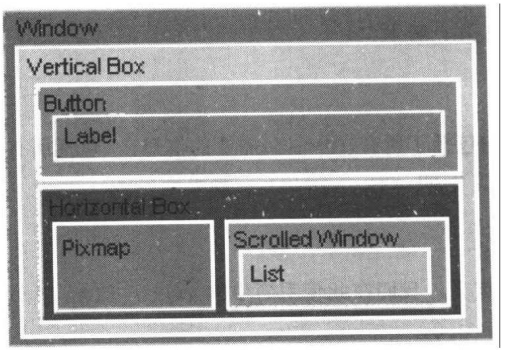

图 1-2 GTK+应用程序中的 widget 层次

在需要把一个操作和一个事件关联的时候，我们必须使用“signal”。特定的 GTK+类方法允许我们把操作和特定的用户函数捆绑在一起。例如，一个按钮有一个“clicked”操作，我们需要在这个操作上捆绑当按钮被按下时执行的函数。每一个信号都是不同的，所以在定义处理函数的时候应该确认使用了正确的函数原型。

简言之，Gnome 为用户提供了一个功能强大，成熟的桌面环境。在这个图形环境的背后

是一个坚实的基础架构。用户可以使用 Gnome 提供的工具在很短的时间内开发出最流行的应用程序。这里我们着重介绍 Gnome 为应用程序开发人员提供的基本技术。主要包括 GTK+，Gnome 和窗口管理器，CORBA 和 ORBit，以及 Gnome 图形结构。

1.2.1 一个入门实例——简单的 Gnome 应用程序

现在，让我们看一个简单的 Gnome 应用程序："Hello World"。它有一个菜单，一个 About 对话框和一个状态条，如图 1-3 所示。

应用程序提供一个下拉式菜单，状态条中给出菜单提示。它自动保存用户定义的菜单加速键，有一个标准的 About 对话框，并且可以自动翻译所有的标准菜单项。

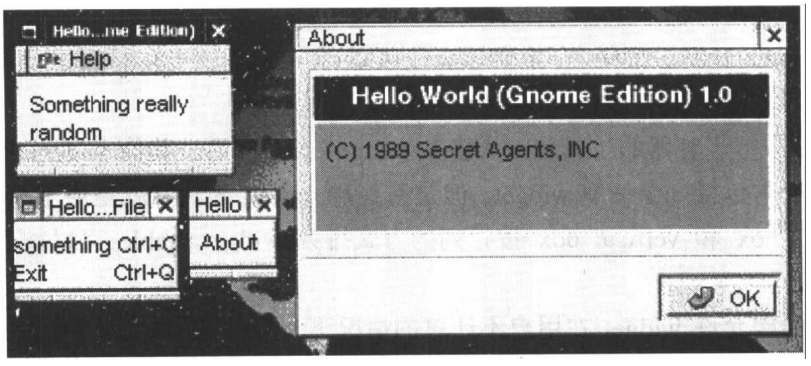

图 1-3 Hello World 程序

源代码如下所示。

```
/*
 *  Hello World (Gnome 版本)
 *
 */
/*  这个 include 语句包括了所有 Gnome 需要的信息，  包括所有
 * Gnome 需要的函数库，例如 gtk, imlib, 等等 ...*/
  #include <gnome.h>

/* 通常由 autoconf  定义，我们在此使用简单的 makefiles */
   #define VERSION "1.0"

/* "callback" 函数 (信号处理) ，它将退出应用程序*/
   static void
   exit_hello(GtkWidget *widget, gpointer data)
   {
        gtk_main_quit ();
   }
```

```
/* 窗口关闭时的 callback 函数 */
 static int
 delete_event(GtkWidget *widget, gpointer data)
 {
     gtk_main_quit ();
     return FALSE; /* false means continue with closing the window */
  }

  /* a callback for the about menu item, it will display a simple "About"
   * dialog box standard to all gnome applications
   */
   static void
   about_hello(GtkWidget *widget, gpointer data)
   {
      GtkWidget *box;
      const char *authors[] = {
           James Bond,
           NULL
       };

      box = gnome_about_new(/*title: */ "Hello World (Gnome Edition)",
               /*version: */VERSION,
               /*copyright: */ "(C) 1999 Secret Agents Inc.",
               /*authors: */authors,
               /*other comments: */
                "An extremely complicated application which "
                "does absolutely nothing useful",
                NULL);
       gtk_widget_show(box);
     }

/* define the menus here */

static GnomeUIInfo file_menu [] = {
          /* some item which is not one of the standard ones, the null
           * would be the callback, however we don't want to really do anything */
           GNOMEUIINFO_ITEM_NONE("Something","Just an item which does
nothing",NULL),
           /* standard exit item */
           GNOMEUIINFO_MENU_EXIT_ITEM(exit_hello,NULL),
           GNOMEUIINFO_END
   };
```

```
static GnomeUIInfo help_menu [] = {
        /* load the helpfiles for this application if available */
        GNOMEUIINFO_HELP("hello_world"),
         /* the standard about item */
        GNOMEUIINFO_MENU_ABOUT_ITEM(about_hello,NULL),
         GNOMEUIINFO_END
 };

 /* define the main menubar */
static GnomeUIInfo main_menu [] = {
        GNOMEUIINFO_MENU_FILE_TREE(file_menu),
         GNOMEUIINFO_MENU_HELP_TREE(help_menu),
        GNOMEUIINFO_END
 };

 /* Our main function */
 int
main(int argc, char *argv[])
 {
        GtkWidget *app; /* pointer to our main window */
        GtkWidget *w; /* pointer to some widget */

        /* initialize gnome */
        gnome_init("hello_world", VERSION, argc, argv);

        /* create main window */
        app = gnome_app_new("hello_world", "Hello World (Gnome Edition)");
        /* connect "delete_event" (happens when the window is closed */
        gtk_signal_connect(GTK_OBJECT(app), "delete_event",
                        GTK_SIGNAL_FUNC(delete_event),
                        NULL);

        /* add the menus to the main window */
        gnome_app_create_menus(GNOME_APP(app), main_menu);

        /* setup appbar (bottom of window bar for status, menu hints and
         * progress display) */
        w = gnome_appbar_new(FALSE, TRUE, GNOME_PREFERENCES_USER);
        gnome_app_set_statusbar(GNOME_APP(app), w);

        /* make menu hints display on the appbar */
        gnome_app_install_menu_hints(GNOME_APP(app), main_menu);
```

```
        /* set up some bogus window contents */
        w = gtk_label_new("Some really really really random\ntext\ngoes
\nhere!");

        /* add those contents to the window */
        gnome_app_set_contents(GNOME_APP(app), w);

        /* show all the widgets on the main window and the window itself */
        gtk_widget_show_all(app);

        /* run the main loop */
        gtk_main();

        return 0;
    }
```

1.2.2 GTK+

GTK+，或者 Gimp Tool Kit，是在 Gnome 应用程序中使用的 GUI 工具包。GTK+ 本身是为 Gimp（GNU Image Manipulation Program）开发的，但是成为了一个具有普遍意义的函数库。GTK+ 依赖于 glib。

GTK+ 开发包中还包括 GDK（Gimp Drawing Kit），它是底层 X Window 系统函数库的一个简化版本。由于 GTK+ 使用 GDK，而不是直接调用 X，GDK 允许 GTK+ 运行在 X 系统以外的其他窗口系统上。GTK+ 和 Gimp 已经提供 Win32 的支持。

GTK+ 为 Gnome 应用程序提供多种特性。

- 一个动态的类型系统。
- 一个用 C 语言编写的对象系统，是通过继承、类型检查和信号、回调机制写成的。类型和对象系统不是特定于 GUI 的。
- 一个使用对象系统编写的 GtkWidget 对象，定义了 GTK+'s 图形组件的实现接口。
- 一个大的实用 GtkWidget 子类 (widgets) 组成了 GTK+ 。
- Gnome 在基本的 GTK+ 集合中添加了一些特定的 widgets。
- Gnome 1.0 是基于 GTK+ 1.2 版本的。所有的 GTK+ 中以 1.2 开头的版本都是与无错误版本兼容的。

GTK+是一个成熟有效的 widget 集合。它不仅提供了最流行的界面，而且还保证了灵活性。GTK+的强大对象系统可以让用户在运行时刻动态查询并修改对象。同时，GTK+还是一个可以扩展的工具包，比如它提供了内置、高层的 widget 来加速 Gnome 应用程序的开发周期。GTK+的 theme 引擎帮助用户定制应用程序的外观。

GTK+的实用性主要表现在以下几个方面。