GTK+2.0 编程范例

——宋国伟　编著——

清华大学出版社

GTK+2.0 编程范例

宋国伟　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

本书以范例程序的形式，由浅入深的引导读者学习开源软件领域和 Linux 平台上最为著名的图形界面开发工具 GTK+2.0 的程序设计，是学习使用 GTK+2.0 编程的良师益友。

本书共 11 章，提供了 70 余个 GTK+2.0 程序范例，可以分两部分。第一部分为第 1 章至第 5 章，主要讲述基础的 GTK+2.0 编程、GTK+2.0 中的常用控件的使用方法、常用的 GTK+2.0 编程技巧及其综合应用等，是初学者入门的导引；第二部分为第 6 章到第 11 章，是前一部分的提高和扩展。主要讲述 GTK+2.0 中新增的复杂控件的使用、文件操作、游戏编程、自定义控件、结合 MySQL 数据库编程、网络编程和 GTK+2.0 中外观启动文件、动态链接库、线程等高级应用。是进一步研究和应用 GTK+2.0 编程，开发实用程序和大型软件的前提。

本书适合利用 GTK+2.0 开发的专业人员，具有 C 语言基础的大中专院校的相关专业的师生以及在 Linux 上开发图形界面的爱好者。

图书在版编目(CIP)数据

GTK+2.0 编程范例/宋国伟编著.－北京：清华大学出版社，2002
ISBN 7-302-06011-8

I.G… II.宋… III.Linux 操作系统－程序设计 IV.TP316.81

中国版本图书馆 CIP 数据核字(2002)第 081653 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.com.cn
http://www.tup.tsinghua.edu.cn
责任编辑：张彦青
印 刷 者：北京国马印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：20　**字数**：469 千字
版　　次：2002 年 11 月第 1 版　2002 年 11 月第 1 次印刷
书　　号：ISBN 7-302-06011-8/TP·3588
印　　数：0001～4000
定　　价：33.00 元

前　言

首先要解释两个概念，GIMP 和 GTK。GIMP(GNU Image Manipulation Program，GNU 图像处理程序)是基于 Linux 平台，能与 Photoshop 相媲美的图像处理软件；GTK(GIMP Tool Kit，GIMP 工具包)最初是 GIMP 的专用开发库，后来发展为 Linux 下开发图形界面应用程序的主流开发工具之一，备受开源软件开发者瞩目。

2002 年 3 月 11 日，GTK+2.0 版正式发布，开发者们自豪地宣称“**GTK+2.0 标志着自由软件领域的图形界面开发工具走向成熟**”。

GTK+2.0 是自由软件，并且是 GNU 工程的一部分。GTK+2.0 的许可协议 LGPL，允许任何开发者使用(包括那些开发中的专利软件)且不收任何费用。GTK+2.0 是当今惟一百分之百零付费的工业级的图形界面开发工具。

在发布 GTK+2.0 软件包同时还发布了 GLib2.0、ATK1.0 和 Pango1.0 这 3 个与其相关的程序库，这些构成了 GNOME2 桌面环境的基础。

GLib2.0 是一个以 C 语言编写的底层核心库，是 GTK+2.0 和 GNOME 的基石。它提供数据结构的 C 语言接口和运行时函数的接口，如事件循环、线程、动态调用和对象系统等。是一个轻量级的实用性非常强的程序库软件包。

Pango1.0 是一个重在国际化的布局和正文翻译库，它为 GTK+2.0 提供核心正文格式化、窗口和字体操作。

ATK1.0 提供友好的界面集，在 ATK 界面的支持下，应用程序或工具软件可以使用其他输入设备。

本书就以 GTK+2.0 为开发工具，向读者全面展示了在 LINUX 平台上开发应用程序的各个侧面，由浅入深地一步一步引导初学者在 LINUX 平台上开发图形界面应用程序。

1. 读者对象

GTK+2.0 是采用 C 语言设计，且本书所有例程都是用纯 C 语言开发的，所以要求读者一定要熟悉、掌握 C 语言，对 Linux 操作系统较熟悉，会用 Linux 命令和 GNOME、KDE 等桌面环境操作，会用 LINUX 下常用的 C 语言编译器 GCC，编译 C 语言程序等。本书适合利用 GTK+2.0 开发的专业人员，具有这些前提的大中院校的相关专业师生以及想在 LINUX 上开发图形界面的 LINUX 爱好者。

2. 运行平台

本书的所有例程都在 RedHat Linux 7.2 操作系统，GNOME2 桌面环境，运用 GCC2.96 编译通过。所有源程序都用 GEDIT 2.0 编辑器编辑，使用 UTF8 格式编码。

正是由于清华大学出版社彭欣编辑的多方努力，才使此书得以出版；我妻子郝春艳女士和我父母对我的写作提供了大力支持，在此对他们表示衷心的感谢。

宋国伟　2002 年 9 月

目　录

第 1 章　窗口与简单控件

本章重点：

开发图形用户界面(GUI)程序的第一步是创建窗口，然后完善窗口的功能，进一步向窗口中添加一些常用的控件，再就是编程使控件响应相关的事件。用 GTK+2.0 来开发 GUI 程序也是如此。

GTK+2.0 提供一种用最短的代码来编写窗口和控件的方法，还有灵活易用的信号/回调函数机制。通过本章的学习，读者能够创建简单的用户界面，编写一般的回调函数，理解信号/回调函数机制。

本章主要内容：

- 如何使用窗口控件
- 如何使用容器控件
- 如何使用按钮控件
- 如何使用标签、图像控件

1.1　最简单的窗口

本节将介绍用 GTK+2.0 创建一个最简单的窗口，并把它显示出来的方法。同时通过这一过程理解 GTK+2.0 程序结构。

实例说明

窗口是 GUI 编程中直接面对用户的操作对象，创建窗口也就成了初学者学习 GUI 编程的第一步。GTK+2.0 提供了非常简便的创建窗口的方法，用它写出的代码在所有 GUI 开发工具中几乎是最短的，也是最容易理解的。

实现步骤

(1) 启动 Linux，进入 GNOME 桌面环境，打开终端输入如下命令：

```
cd ~
mkdir ourgtk
cd ourgtk
mkdir 1
cd 1
mkdir base
cd base
```

创建工作目录，并进入此目录开始编程。

说明：首先进入用户工作目录，创建一个学习 GTK+2.0 总的工作目录 ourgtk，接着创

建本章的工作目录 1，然后再创建本节的工作目录 base，进入此目录开始工作。

(2) 打开编辑器(GEDIT2.0 见前言)。输入如下代码，以 base.c 为文件名保存到当前目录(base)下：

```
/* 最简单的窗口base.c */
#include <gtk/gtk.h>
int main ( int argc , char* argv[])
{
GtkWidget *window;
    gtk_init(&argc,&argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_widget_show(window);
    gtk_main();
return FALSE;
}
```

(3) 编辑 Makefile 输入如下代码：

```
CC = gcc
all:
    $(CC) -o base base.c 'pkg-config --cflags --libs gtk+-2.0'
```

注意： $(cc)前面不是空格，而是一个<tab>键，如果用空格代替编译时会出问题。

以 Makefile 为文件名保存到 base 目录下。

(4) 在终端中执行 make 命令开始编译；

(5) 编译结束后，执行命令./base 即可运行此程序，运行结果如图 1.1 所示：

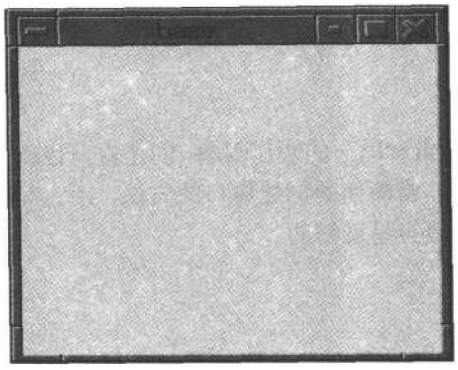

图 1.1 最简单的窗口

实例分析

(1) 设置 include 文件与声明主函数

首先是#include <gtk/gtk.h> 语句，这是每个 GTK+2.0 程序都要包含的头文件。然后是函数声明 int main (int argc , char* argv[])，这是标准的 C 语言主函数的声明。

(2) 声明变量

GtkWidget *window; 声明了窗口控件的指针。其中 GtkWidget 是 GTK+2.0 控件类型，

几乎所有的 GTK+2.0 控件都用这一类型声明；window 是变量名，它与变量类型无关。完全可以给它起一个诸如 asd21i9 这样的名字，不过这只会令程序代码更难读，所以最好起一个易于理解的变量名，比如 window。

(3) 初始化 GTK+2.0 的命令行参数

函数 gtk_init(&argc,&argv); 初始化命令行参数。这在 GTK+2.0 程序中是必需的，不管您的设计中是否使用到命令行参数，都需要用这一函数来初始化。

(4) 创建窗口

代码行 window = gtk_window_new(GTK_WINDOW_TOPLEVEL); 用来创建窗口。函数 gtk_window_new 创建一个窗口并返回这个窗口的控件指针，在这里这个指针的值赋给了变量 window；参数 GTK_WINDOW_TOPLEVEL 指明窗口的类型为最上层的主窗口，它最常用。还可以取另一个值 GTK_WINDOW_POPUP 指明窗口的类型为弹出式的无边框的窗口。

(5) 显示窗口

代码行 gtk_widget_show(window); 用来显示上一步创建的窗口。函数 gtk_widget_show 是用来显示控件的，它没有返回类型，参数是要显示的控件的指针，在这里是窗口 window，所以窗口就显示出来了。

(6) 主事件循环

最后这个函数 gtk_main(); 是最关键的，它是 GTK+2.0 的主事件循环，每一个 GTK+2.0 程序都要有一个，否则程序就无法运行。所谓事件循环是指 GUI 程序运行时等待来自外部用户发出的事件，如键入或鼠标移动等，GTK+2.0 将这些事件包装成信号，用户再根据信号的功能编写相应的回调函数来处理这些事件。这段代码中并未写回调函数，所以这只是一个死循环，不做任何反应。

代码最后返回逻辑值 FALSE，它相当于整型的 0。不写这一行是完全可以的，但写这行代码更能体现程序的完整性和可读性。

(7) 编译运行

以上代码完全可以用下面命令行直接编译：

```
gcc -o base base.c 'pkg-config --cflags --libs gtk+-2.0'
```

但在终端上输入这么一长串的命令非常繁琐且容易出错，更糟糕的是如果长时间不用的话还可能忘记这串命令。幸好 Linux 提供了 make 工具，这样按上面步骤编辑好 make 的配置文件 Makefile，在命令行中直接输入命令 make 就可以编译了，编译结束后输入命令./base 就可运行这个程序了。(注意：base 前面一定要加上点和斜杠，表示在当前目录下运行，否则的话终端找不到要运行的程序。)

读者还会注意到命令行中的'pkg-config --cflags --libs gtk+-2.0'，它向编译器指出了包含文件的路径，动态链接库路径和要链接哪些动态链接库。pkg-config 是 GTK+2.0 和 GNOME2 系统必备的软件包配置和管理工具，可以在命令中直接运行。

至此，我们完成了最简单的变量的声明、窗口的创建、显示，了解了 GTK+2.0 程序的一般结构、GTK+2.0 程序中的两个关键步骤：gtk_init 和 gtk_main。这些都是 GTK+2.0 程序中必需的，为下面进一步学习 GTK+2.0 编程打下基础。

1.2 完善窗口的功能

本节将介绍如何完善窗口的功能，为窗口或控件添加回调函数，对窗口进行相应的功能设置，使它能真正的退出，更加实用。

实例说明

上节中的示例运行时，单击关闭按钮，窗口会自动关闭，但终端中的提示符却不见了。其实完全可以关闭终端窗口，再开启一个。不过最好的办法是按下 Ctrl+C 键，这样提示符就出现了。

为什么会出现这种情况呢？上面示例中只调用了主事件循环 gtk_main，而未对来自用户的信号做任何处理，所以即使窗口关闭了，循环仍在执行，程序并未真正退出。按下 Ctrl+C 键后，系统向此程序发出了中断信号，程序才真正地终止执行了。

本节示例以解决这一问题为目的，更进一步完善窗口的功能。

实现步骤

(1) 打开终端输入如下命令：

```
cd ~/ourgtk/1
mkdir window
cd window
```

创建工作目录，进入此目录开始编程。

(2) 打开编辑器，输入以下代码，以 window.c 为文件名保存到当前目录下：

```
/* 完善窗口的功能 window.c */
#include <gtk/gtk.h>

int main ( int argc , char* argv[])
{
    GtkWidget *window;
    gtk_init(&argc,&argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    g_signal_connect(G_OBJECT(window),"delete_event",
            G_CALLBACK(gtk_main_quit),NULL);
    gtk_window_set_title(GTK_WINDOW(window),"一个功能完善的窗口");
    gtk_window_set_default_size(GTK_WINDOW(window),500,100);
    gtk_window_set_position(GTK_WINDOW(window),GTK_WIN_POS_CENTER);
    gtk_widget_show(window);
    gtk_main();
    return FALSE;
}
```

(3) 编辑 Makefile 输入以下代码：

```
CC = gcc
all:
```

```
$(CC) -o window window.c 'pkg-config --cflags --libs gtk+2.0'
```

(4) 在终端中执行 make 命令开始编译；

(5) 编译结束后，执行命令./window 即可运行此程序，运行结果如图 1.2 所示。

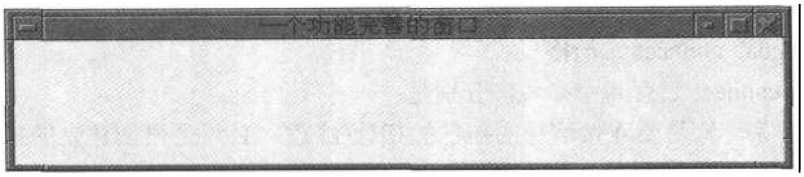

图 1.2　功能完善的窗口

实例分析

(1) 信号与回调函数

GTK+2.0 采用了一种信号/回调函数机制来处理窗口外部传来的事件、消息或信号。即先为窗口或控件定义一系列信号，在编程中引用信号名称为窗口或控件添加回调函数，当信号发生时，程序自动调用为信号连接的回调函数。

窗口的信号有很多，如“key_press_event”在按键时发生，“foucs”在获得焦点时发生等等。这里主要介绍“delete_event”，这一信号在窗口关闭时发生。

(2) 退出程序

退出 GTK+2.0 程序要调用 gtk_main_quit();函数，它的功能就是退出主循环，也就是结束程序的运行。一般情况下当窗口关闭时程序就退出了，但 GTK+2.0 并不主动处理退出程序。这就需要为上面介绍的“delete_event”信号连接回调函数。

(3) 为信号连接回调函数

为窗口或控件加回调函数有两种方式：

一种方式是直接调用已有函数(如退出函数 gtk_main_quit)，在窗口或控件创建完成后直接引用 g_signal_connect 宏，代码如下所示：

```
g_signal_connect(G_OBJECT(window),"delete_event",
        G_CALLBACK(gtk_main_quit),NULL);
```

另一种方式是先定义好回调函数，在窗口或控件创建完成后再引用 g_signal_connect 宏，代码如下所示：

```
/* 声明回调函数 on_delete_event */
void    on_delete_event (GtkWidget *widget,GdkEvent* event,gpointer data)
{
    gtk_main_quit();
    return FALSE;
}
......
/* 在主函数中为窗口的"delete_event"信号加回调函数 */
......
g_signal_connect(G_OBJECT(window),"delete_event",
        G_CALLBACK(on_delete_event),NULL);
......
```

在 GTK+1.2 中经常用第二种方式来退出程序运行，在 GTK+2.0 中就完全可以不用这种方式，采用第一种方式更简单明了。这种方式的好处是：可以在 on_delete_event 函数中加入其他代码来处理程序开始时遗留的问题(如释放已分配的内存、保存程序配置等)，还可以询问用户是否真正退出程序运行。

(4) g_signal_connect 宏的格式

g_signal_connect 宏有 4 个参数，分别是：

连接的对象，就是要连接信号的控件的指针(注意：必须是已创建完的控件的指针)，需要用 G_OBJECT 宏来转换，如本例中的 G_OBJECT(window)；

信号名称，就是要连接的信号名称，为字符串形式，用双引号引起来。不同的控件拥有的信号名称是不一样的，如本例中的窗口控件的信号："delete_event"；

回调函数，指信号发生时调用的函数，这里只用到函数名称，需要用 G_CALLBACK 宏来转换一下，如本例中的 G_CALLBACK(gtk_main_quit)；

传递给回调函数的参数，它的值类型应该为 gpointer，如果不是这一类型可以强制转换，如果没有参数则为 NULL。这里只能传递一个参数，如果有多个参数，可以先将多个参数定义为一个结构，再将此结构作为参数传递过来。

注意： gpointer、gchar、gint、gboolean 等类型是在 GTK+2.0 的基础库 GLIB2.0 中定义的，是对 C 语言基础数据类型的包装。它渗透到 GTK+2.0 编程的每一个角落，读者一定要细致地研究一下，这对进一步学好 GTK+2.0 编程很有好处。

(5) 回调函数的格式

不同的控件的信号不同，不同的信号的回调函数的格式也不同。有一个规律，即多数回调函数是没有返回类型的，名称可以自定义，最好能表达一定的意思，参数有多个，第一个参数是调用此回调函数的控件对象指针，最后一个参数是用户传递给此回调函数的参数，而且固定为 gpointer 类型。如上面介绍的 on_delete_event 函数表示当“delete_event”信号发生时执行，它的第 1 个参数 widget 是指窗口控件的指针，第 2 个参数 event 是指事件类型，第 3 个参数 userdata 是指用户传递的参数。

(6) 改变窗口外观的几个函数

设定窗口的标题：

```
gtk_window_set_title(window,const gchar* title);
```

设定窗口的默认宽高：

```
gtk_window_set_default_size(window,int width,int height);
```

设定窗口的位置：

```
gtk_window_set_position(window,GtkWindowPosition position);
```

其中 position 可以取如下值：

```
GTK_WIN_POS_NONE      不固定
GTK_WIN_POS_CENTER    居中
GTK_WIN_POS_MOUSE     出现在鼠标位置
GTK_WIN_POS_CENTER_ALWAYS        窗口改变尺寸仍居中
```

GTK_WIN_POS_CENTER_ON_PARENT　　居于父窗口的中部

更多的函数见 GTK+2.0 的 API 参考手册。

运行这个程序时会发现，窗口的标题是中文的，窗口的位置处于屏幕中央，且尺寸也变了。更可喜的是关闭窗口后，终端上的提示符马上出现了。

本节的示例主要研究了信号和回调函数的一般格式和用法，这是编写 GTK+2.0 程序的关键所在，读者还应加强这方面的理解。

1.3　Hello 程序

本节将介绍如何向窗口中添加按钮控件，为按钮控件加回调函数——实现一个简单的计数器，并在窗口中显示文字。

实例说明

一般的程序设计课程都以一个 Hello 程序作为第一个示例，在 GTK+2.0 中也很容易做到。首先创建一个窗口，在窗口中添加一个按钮，单击按钮，使之做出反应，在终端上显示一行信息。

实现步骤

(1) 打开终端输入如下命令：

```
cd ~/ourgtk/1
mkdir hello
cd hello
```

创建工作目录，并进入此目录开始编程。

(2) 打开编辑器，输入以下代码，以 hello.c 为文件名保存到当前目录下：

```
/* hello程序  hello.c */
#include <gtk/gtk.h>
gint count=1;
void
on_button_clicked   (GtkWidget *button,gpointer userdata)
{
    g_print("你好，这是Hello功能的测试.   ");
    //g_print("Hello . This is a test . ");
    g_print("%d\n",(gint)userdata);
    //g_print("%d\n",count);
    count = count + 1 ;
}
int main ( int argc , char* argv[])
{
    GtkWidget *window;
    GtkWidget *button;
    gtk_init(&argc,&argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

```
    g_signal_connect(G_OBJECT(window),"delete_event",
            G_CALLBACK(gtk_main_quit),NULL);
    gtk_window_set_title(GTK_WINDOW(window),"Hello 功能实现");
    gtk_window_set_default_size(GTK_WINDOW(window),500,100);
    gtk_window_set_position(GTK_WINDOW(window),GTK_WIN_POS_CENTER);
    gtk_container_set_border_width(GTK_CONTAINER(window),40);
    button = gtk_button_new_with_label("按下此按钮会在终端上显示一行信息
");
    g_signal_connect(G_OBJECT(button),"clicked",
            G_CALLBACK(on_button_clicked),(gpointer)count);
    gtk_container_add(GTK_CONTAINER(window),button);
    gtk_widget_show(button);
    gtk_widget_show(window);
    gtk_main();
    return FALSE;
}
```

(3) 编辑 Makefile 输入以下代码：

```
CC = gcc
all:
    $(CC) -o hello hello.c 'pkg-config --cflags --libs gtk+2.0'
```

(4) 在终端中执行 make 命令开始编译；

(5) 编译结束后，执行命令./hello 即可运行此程序，运行结果如图 1.3 所示。

图 1.3 HELLO 程序

实例分析

(1) 按钮控件

这里介绍 GTK+2.0 中常用的普通按钮控件。GTK+2.0 控件是统一用 GtkWidget 类型来管理的，按钮控件也不例外。我们可以用 gtk_button_new 来创建一个不显示内容的空按钮，还可以用 gtk_button_new_with_label 来创建一个显示文字的按钮，本示例中用的是后者，相信读者一看就知道。

注意代码书写顺序，先定义，再创建，再加回调函数，最后显示。

看到这里读者一定会明白 GTK+2.0 的函数名称特色：以 gtk 开头，用下划线连接；第二个单词表示控件的类型，如：button、window 等；再后面的相关的单词表示要做的动作，如：new_with_label(新建并带标签)，set_default_size(设定默认的尺寸)等。使读者基本能见其名而知其含义，也非常容易掌握。

(2) 容器

GTK+2.0 中的控件的摆布采用了容器(container)这一概念，即所有 GTK+2.0 控件分成两种，一种是能容纳其他控件的容器，另一种是不能容纳其他控件的非容器控件。容器控件又分成能容纳多个控件的容器和只能容纳一个控件的容器。

窗口控件也是一种容器，它可以把按钮容纳进来。如何操作呢？GTK+2.0 提供了与容器相关的操作函数。以 gtk_container 开头，函数 gtk_container_add 的功能是将另一控件加入到容器中来。它的第一参数是 GtkContainer 型的指针，这就需要将窗口控件指针用宏 GTK_CONTAINER 转换一下，即 GTK_CONTAINER(window)。它的第二参数是要容纳的控件的指针，即 button。另一个常用到的与容器相关的函数是 gtk_container_set_border_width。它用来设定容器边框的宽度，格式如例中所示，宽度的单位是像素，可以根据需要自行设定。如不设定窗口边框的宽度，其默认值是 0，这样的话按钮的显示不明显。

(3) 按钮的回调函数

创建完按钮后，我们给按钮的单击信号“clicked”加了一个自定义的回调函数。这个命名为 on_button_clicked 的函数，其格式一般如下：

```
void 函数名 (GtkButton* button,gpointer userdata)
```

在这个回调函数中，使用了 g_print 函数。它的格式和 C 语言中最常用的 printf 函数格式一样，而且功能也一样，都是向输出设备输出格式化字符串。

代码中我们还定义了一个静态计数器——整型变量 i，即每执行一次这个函数(每单击一次按钮)其值自动加一，这样每次输出的数字也就都不一样了。

运行此程序，单击按钮，终端上就会输出“你好，这是 Hello 功能的测试．1”，再按一下 1 就会变成 2，如此继续直到关闭窗口。如果您的终端窗口不支持中文，将代码中的中文输出行加上注释，去掉另两行(英文输出)的注释则会显示英文输出结果。

编完这个示例可能有些读者会很失望，因为只在终端中输出信息，如果能直接弹出消息框就好了。不过别着急，第 4 章将详细介绍各种消息框的使用。

1.4　用盒状容器排列按钮

本节将介绍 GTK+2.0 中能容纳多个控件的盒状容器，以及如何灵活使用盒状容器控件，如何在盒状容器中排放多个控件和设置它们的排放属性。

实例说明

当继续向窗口中添加按钮时会发现程序又出错了，这是因为窗口只能容纳一个控件。如何容纳多个控件呢？上节我们知道容器又分成两种，窗口属于只能容纳一个控件的容器，如果向窗口中加一个能容纳多个控件的容器，再向此容器中添加其他别的控件，就可以实现这一目标。

GTK+2.0 中能容纳多个控件的容器主要有盒状容器(GtkBox)、格状容器(GtkTable)、按钮盒(GtkButtonBox)、分隔面板(GtkPanel)、固定布局(GtkFixed)、工具栏(GtkToolbar)等，

其中最常用的是盒状容器和格状容器。

盒状容器是一种按一定顺序和方向紧密排列多个控件的容器。在盒状容器(GtkBox)基础上又分成纵向盒状容器(GtkVBox)和横向盒状容器(GtkHBox)，两者除了创建和排列控件的方式不一样外，其使用方式是一模一样的。

我们还应注意的是容器中还能容纳其他容器，这形成了用 GTK+2.0 创建复杂用户界面的基础。一定要把容器间容纳与被容纳关系理顺。

实现步骤

(1) 打开终端输入如下命令：

```
cd ~/ourgtk/1
mkdir pack
cd pack
```

创建工作目录，并进入此目录开始编程。

(2) 打开编辑器，输入以下代码，以 pack.c 为文件名保存到当前目录下：

```
/* 用盒状容器排列按钮 pack.c */
#include <gtk/gtk.h>
int main ( int argc , char* argv[])
{
    GtkWidget *window;
    GtkWidget *box;
    GtkWidget *button;
    gchar *title = "排列按钮" ;
    gtk_init(&argc,&argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    g_signal_connect(G_OBJECT(window),"delete_event",
            G_CALLBACK(gtk_main_quit),NULL);
    gtk_window_set_title(GTK_WINDOW(window),title);
    gtk_window_set_position(GTK_WINDOW(window),GTK_WIN_POS_CENTER);
    gtk_container_set_border_width(GTK_CONTAINER(window),20);
    box = gtk_hbox_new(FALSE,0);
    gtk_container_add(GTK_CONTAINER(window),box);
    button = gtk_button_new_with_label("按钮一");
    gtk_box_pack_start(GTK_BOX(box),button,TRUE,TRUE,3);
    button = gtk_button_new_with_label("按");
    gtk_box_pack_start(GTK_BOX(box),button,FALSE,FALSE,3);
        button = gtk_button_new_with_label("钮");
    gtk_box_pack_start(GTK_BOX(box),button,FALSE,FALSE,3);
    button = gtk_button_new_with_label("二");
    gtk_box_pack_start(GTK_BOX(box),button,FALSE,FALSE,3);
    gtk_widget_show_all(window);
    gtk_main();
    return FALSE;
}
```