

Glade Tutorial

By Paul Hogan

p.j.hogan@open.ac.uk - pachjo@hotmail.co.uk

October 2006



翻译: webdisk008

目 录

ii 关于术语的说明	1
1 Glade 起点	2
2 Gedit 起点	6
3 Python 运行 第一次	8
4 Glade 配置Window1	9
5 Gedit 连接Window1 的destroy 信号	12
6 Python 运行 第二次	13
7 Glade 添加一个按钮	13
8 Python 运行 第三次	16
9 Glade 配置按钮部件	16
10 Python 运行 第四次	19
11 Glade 完成GUI	20
12 Python 运行 第五次	21
13 Gedit 完成Python 代码	22
14 Python 运行 第六次	23
15 结束语	24

i 简介

我写这篇教程为的是告诉大家我是怎样学习和领会如何在 Linux 环境下使用 Glade、Gedit 和 Python 语言来创建一个 GUI（Graphical User Interface，图形用户界面）应用程序的。

在 Windows 下我发现 GUI 应用的创建方式和是完全不同的，耗费时间并且非常令人沮丧，比如使用 Microsoft Access 等开发环境。

术语也令人困惑，但是这种个别部分的转变是很容易克服的。

有大量的帮助文档会教你如何安装所需的应用程序，所以在这篇教程中我就不再包括这部分了。然而我还会告诉你，想要贯通这篇教程需要安装如下的软件包：

- Python 2.4 或者与之相关联的函数库
- Glade 或者 Glade Gnome 以及相关关联的函数库
- Gedit 文本编辑器

为了方便，我在不同的桌面上都配置好了我所使用的开发环境。这样我们在创建我们的应用时可以方便的选择每个运行中的开发工具。因此，现在我们在每个桌面上打开 Glade、Gedit、Python 和一个控制台（console，终端）。（用四个桌面分别打开其一）

这篇教程所使用的方法是从 GUI 设计到编辑 Python 代码再到运行应用程序的每个开发阶段的循环。

因此，我们从 Glade 到 Gedit 再到控制台开始，然后再回到 Glade 并且按这种方式继续循环一直到我们完成我们的应用程序。

ii 关于术语的说明

Linux 中的 GUI 有两种主要的对象类型，widgets（窗体小部件）和 containers（容器）。

Widgets 对象是窗口、按钮、标签、复选框等等。**Containers** 是容纳这些小部件的对象。

Linux 中的 GUI 有 signals 和 callbacks（callback 函数）。（信号和回调）

Signals 是在响应一个事件（Event）时产生的，比如点击一个按钮的事件，这将会产生一个“on_clicked” signal（信号）。

Callbacks 是用来响应 signals（信号）的函数。

在 Windows 下这些被称为事件和事件处理程序。

1 Glade 起点

1.1 选择 Glade desktop

1.2 你应当看一下下边的详细组成，如果你没有相同的布局，那么单击标题为“Glade: <untitled>”的对话框的 View（视图或查看）菜单并且选中除了“Show Clipboard（显示剪贴板）”的所有选项。

1.3 把全部这四个窗口打开大概是个不错的做法，因此我们就照这样做。

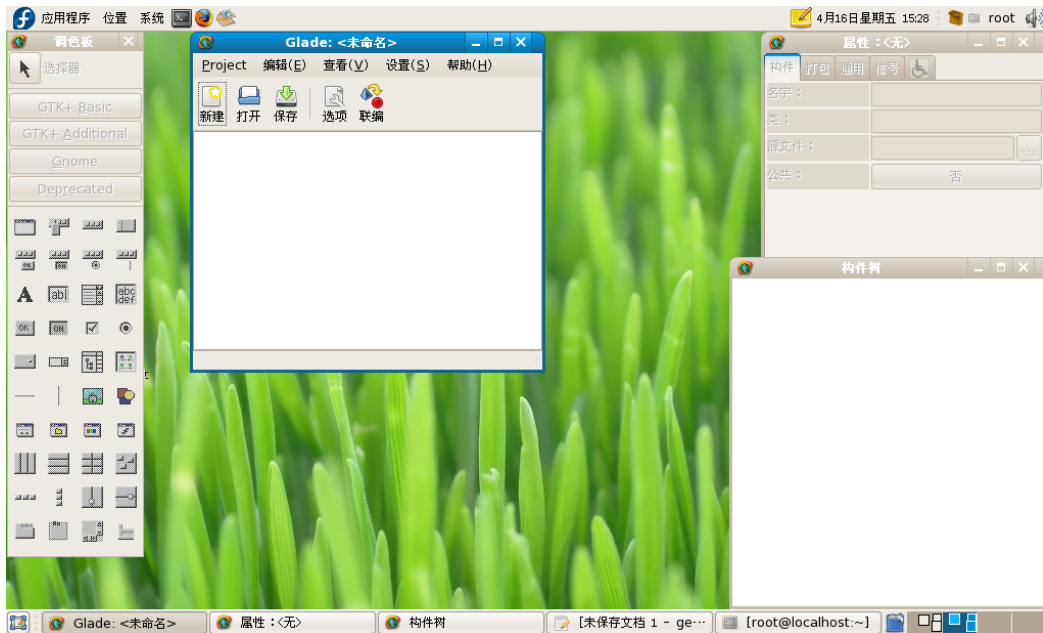


图 1 Glade 的整体布局

1.4 主窗口

1.4.1 下图所示的这个窗口就是主窗口，它允许你打开、保存其他文件并为你的工程设置选项。它还可以在工具栏下面的区域中把你的应用程序中使用的所有 widgets 以列表的形式列出。



图 2 主窗口

1.5 调色板窗口

1.5.1 调色板窗口允许你从中选择 widgets 和 containers 用于你的应用程序。下图中右边的调色板窗口有一个 Gnome 按钮，这个按钮只有安装了 glade gnome 后才会显示。

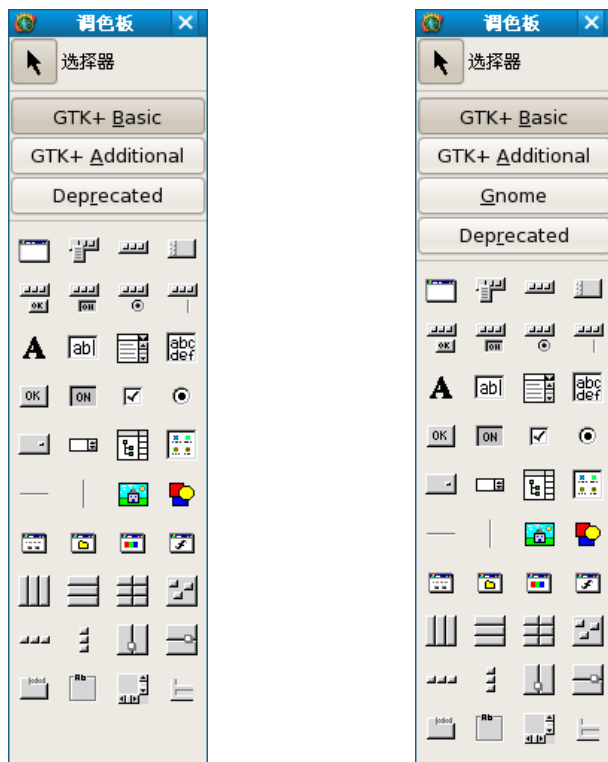


图 3 两种类型工程中的调色板 GTK+（左） GNOME（右）

1.6 构件树（Widget Tree）窗口

1.6.1 这个窗口以（树状的）垂直视图显示了在你的应用程序中使用的全部 widgets 和 containers。

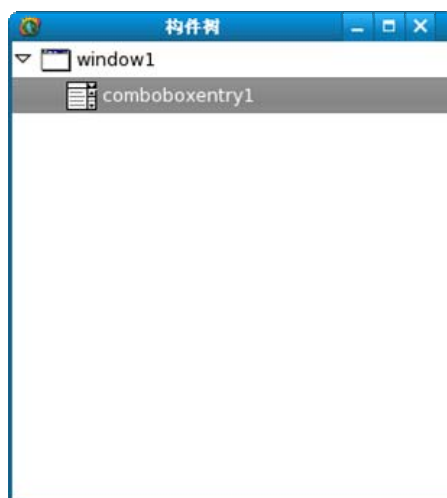


图 4 构件树窗口

1.7 属性窗口

1.7.1 这个窗口允许你配置你的 widgets 和 containers 的各种设置。



图 5 属性窗口

1.8 点击主窗口上的“New（新建）”按钮显示新建工程对话框。



图 6 新建工程对话框

1.9 点击“New GTK+ Project”按钮。

1.10 点击主窗口上的“Save（保存）”按钮显示工程选项对话框。通过这一步我们设置存储我们应用程序的所有文件的所需目录。



图 7 项目选项对话框（初始）

1.11 更改工程目录域的最后一项，把“项目 1”改为“glade-tut”。

1.12 注意：工程名和程序名也变成了“glade-tut”。



图 8 项目选项对话框（修改工程目录）

1.13 点击“确定”按钮保存工程。

1.14 查看一下工程目录中的内容，看看你刚刚创建了什么。两个文件已经被创建了：
glade-tut.glade 和 glade-tut.glade.p。



图 9 工程目录下生成的文件

.glade 文件是 XML 源文件，我们将使用 Python 语言来编写它；不用管这个文件，因为我们对它做的每件事情都将被 Glade 完成。

.glade.p 文件是 Glade 工程文件，同样的，我们对它做的每件事情都将被 Glade 完成。

1.15 让我们回到 Glade 桌面，点击调色板窗口上的“GTK+ Basic”按钮。

1.16 点击调色板窗口上的“窗体”按钮（，控件区域第一排左数第一个）。当你把鼠

标悬停在某个按钮上时，你将会看到一个工具提示，它将帮助你定位到正确的按钮。

1.17 创建了一个新的窗体，显示的标题为“window1”。



图 10 新建的窗体

1.17.1 主窗口将会在它的显示区域显示“window1”。

1.17.2 构件树窗口将会显示“window1”。

1.17.3 属性窗口将会显示“window1”的属性。

1.18 点击主窗口工具栏上的“Save（保存）”按钮将变更保存。

1.19 以上就是到目前为止我们创建一个基本的没有什么令人兴奋的 GUI 需要做的全部事情。下一节将详细讲解如何创建 Python 文件来满足运行我们的应用程序的需要。

2 Gedit 起点

2.1 选在 Gedit 所在的桌面并选择 Gedit 窗口。

2.2 在做任何事情之前，用 Gedit 在存放我们的 Glade 文件的同一目录下保存 Python 文件，文件名我们命名为 glade-tut.py。这一操作意味着 Gedit 会帮助我们在编写 Python 代码时根据 Python 语法来把一些关键字高亮或用彩色显示。

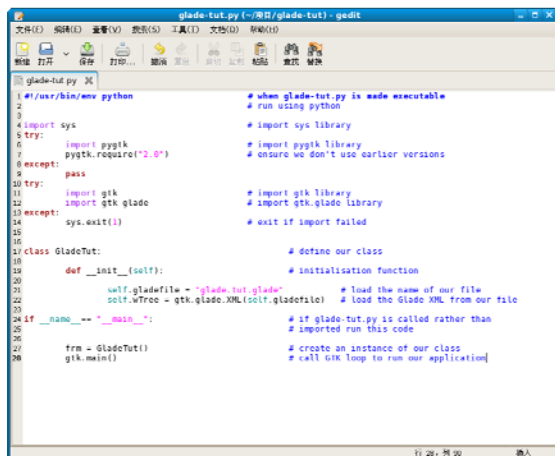


图 11 Gedit 用于代码编辑

2.3 输入下面的代码到刚刚新建的 glade-tut.py 文档里。

```
#!/usr/bin/env python                                # when glade-tut.py is made executable
                                                    # run using python

import sys                                           # import sys library
try:
    import pygtk                                    # import pygtk library
    pygtk.require("2.0")                            # ensure we don't use earlier versions
except:
    pass
try:
    import gtk                                      # import gtk library
    import gtk.glade                                # import gtk.glade library
except:
    sys.exit(1)                                     # exit if import failed

class GladeTut:                                     # define our class

    def __init__(self):                             # initialisation function

        self.gladefile = "glade-tut.glade"          # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile)  # load the Glade XML from our file

if __name__ == "__main__":                          # if glade-tut.py is called rather than
                                                    # imported run this code

    frm = GladeTut()                                # create an instance of our class
    gtk.main()                                       # call GTK loop to run our application
```

2.4 这不是 Python 教程，所以我仅仅解释在使用 Glade 时用到的一小部分，你还需要扩展阅读大量的 Python 教程。

2.4.1 import 声明用来装入所需要的库文件。

2.4.2 GladeTut 是我们定义的一个类，目前它仅包含 “__init__” 函数。这个函数在创建这个类的一个实体时会被调用。我们的 Glade 文件被加载，它定义了我们的 GUI。

2.4.3 “if 块” 用来检查这段代码是否正在被 Python 调用或者作为一个 import 声明的一部分。如果 Python 正在调用它，那么一个 GladeTut 类的实体将被创建，并且 GTK.Main() 函数也会被调用来运行我们的应用程序。

2.5 点击 Gedit 的 “Save (保存)” 按钮。

2.6 现在我们可以运行它，看看我们的应用程序是什么样子的，在下一节将详细讲解

我们如何运行我们的应用程序，以便查看它当前的样子。

3 Python 运行 第一次

3.1 选择 Linux 终端（控制台）所在桌面并且点击选中终端窗口。

3.2 进入到工程文件所在目录，我的命令是：

```
cd /root/项目/glade-tut/
```

3.3 输入如下命令运行我们的应用程序。

```
python glade-tut.py
```

3.4 和下图一样，你将看到一个小窗口出现了，但你的窗口可能出现在你桌面上的不同位置。

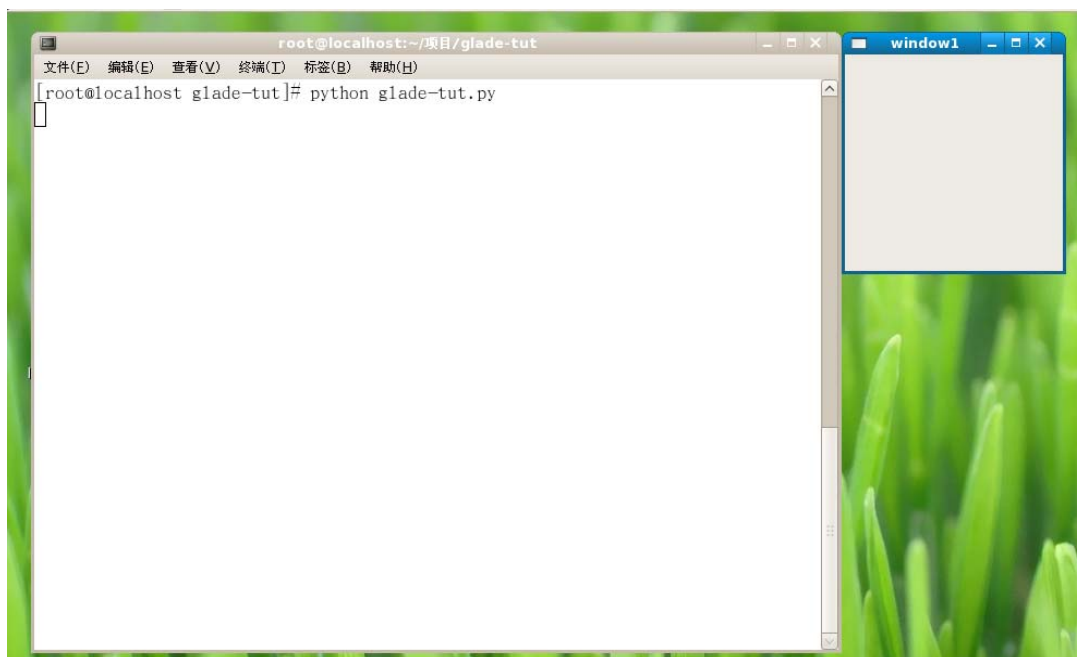


图 12 第一次运行 glade-tut.py

3.5 现在，我们已经在运行一个什么功能也没有的应用程序了，但是如果你玩儿一下你会发现它具有全部的 GUI 功能性。

3.6 点击这个小窗口右上角的“×”，关闭这个小窗口。

3.7 在这里，来看我们遇到的第一个问题，小窗口已经关闭了，但是我们的终端却没有马上返回到提示符状态！这是因为我们还没有告诉 GTK 要停止运行，因此我们按下键盘上的 Ctrl+C 组合键返回到提示符状态。

3.8 现在，我们已经返回到提示符状态，我们完成了如介绍所说的我们的第一次开发循环；也就是说，从 Glade 到 Gedit，再到终端。现在我们返回到 Glade 去修改我们的应用程序，让它在我们点击“×”时能够顺利的关闭。

4 Glade 配置 Window1

4.1 选择 Glade 所在的桌面。

4.2 确保属性窗口显示 window1 的属性。在 window1 上面的任何地方单击鼠标右键，在弹出菜单中选择 window1，然后选择“Select（选择）”。

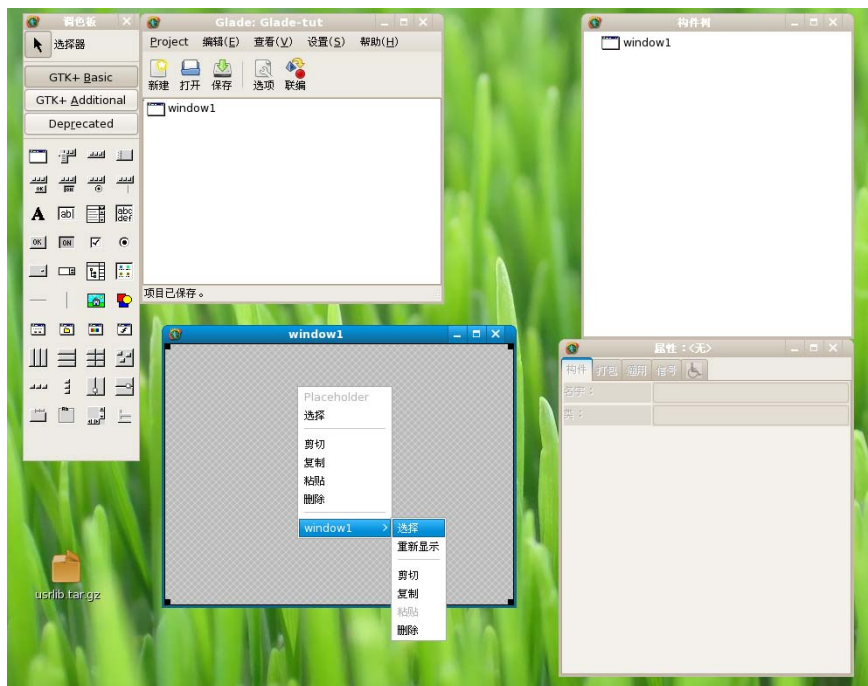


图 13 选择 window1 以查看其属性

4.3 这时，属性窗口将会显示 window1 的属性。



图 14 选中 window1 后的属性窗口

4.4 为了节省一点儿时间，在一次运行前我们就先少修改几个设置；我将解释它们是什么以及它们是做什么用的。首先，点击 Widget（部件）选项卡。

4.4.1 更改“Title（标题）”为 Glade-Tut，这将会把窗口的标题从 window1 改成 Glade-Tut。

4.4.2 更改“Position（位置）”为 Centre，这将会使我们的窗口出现在我们的桌面的中心位置。

4.4.3 更改“Resizable（调整大小）”为“No（否）”。

4.5 点击 Common（通用）选项卡。

4.5.1 点选 Width（宽度）复选框使之在方框中打钩，并输入窗口宽度为 300。这将设置我们窗口的宽度。

4.5.2 点选 Height（高度）复选框使之在方框中打钩，并输入窗口高度为 200。这将设置我们窗口的高度。

4.6 点击 Signals（信号）选项卡。

4.6.1 点击 Signals（信号）文本框右侧的“...”按钮打开信号选择对话框。

4.6.2 向下滚动右侧的滑动条直到你看到以 GObject Signals（GObject 信号）开头的选项，选择它下面的“destroy”选项，然后点击 OK（确定）按钮。

4.6.3 返回到了信号选项卡，点击 Add（添加）按钮。一个条目将会出现在 Signals（信号）选项卡的显示区域。信号列将会显示“destroy”，处理列显示“on_window1_destroy”。

4.7 点击 Glade 主窗口上的“Save（保存）”按钮。

4.8 你的 window1 属性窗口现在将如下图所示。

4.8.1 部件选项卡（Widget tab）



图 15 修改后的 window1 部件选项卡

4.8.2 通用选项卡（Common tab）



图 16 修改后的 window1 通用选项卡

4.8.3 信号选项卡（Signals tab）

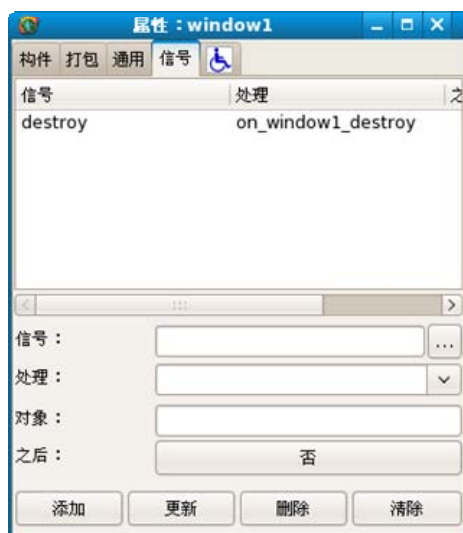


图 17 修改后的 window1 信号选项卡

4.9 信号选项卡（Signals tab）需要一点点解释。回到“关于术语的说明”那一节，你当时应该读了关于信号（signals）和回调（callbacks）的说明。信号选项卡（Signals tab）就是定义你想要你的 Python 代码响应的地方。

这里，我们已经设置了一个简单的信号来响应 window1 的 destroy 时间。当窗口被关闭时，destroy 信号被发出，并且回调函数通过调用 on_window1_destroy 来响应这个信号。

4.10 现在我们将在我们的 glade-tut.py 文件里编写回调函数 on_window1_destroy 的代码。

5 Gedit 连接 Window1 的 destroy 信号

5.1 选择 Gedit 所在桌面，选中 Gedit 窗口。

5.2 修改 “__init__(self)” 函数如下所示。

```
#!/usr/bin/env python                                # when glade-tut.py is made executable
                                                    # run using python

import sys                                            # import sys library
try:
    import pygtk                                    # import pygtk library
    pygtk.require("2.0")                            # ensure we don't use earlier versions
except:
    pass
try:
    import gtk                                      # import gtk library
    import gtk.glade                                # import gtk.glade library
except:
    sys.exit(1)                                     # exit if import failed

class GladeTut:                                     # define our class

    def __init__(self):                             # initialisation function

        self.gladefile = "glade-tut.glade"          # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile)   # load the Glade XML from our file

        dic = {"on_window1_destroy" : gtk.main_quit } # create dictionary for connections

        self.wTree.signal_autoconnect(dic)          # connect our code to the signal

if __name__ == "__main__":                          # if glade-tut.py is called rather than
                                                    # imported run this code

    frm = GladeTut()                                # create an instance of our class
    gtk.main()                                       # call GTK loop to run our application
```

5.3 黄色高亮部分的第一行作用是创建一个字典来保存我们想要做的连接的细节。第二行调用了 `signal_autoconnect()` 函数进行连接。

当窗口被关闭时，`destroy` 信号发出，`on_window1_destroy` 回调函数被调用；就像你看到的一样，将会之星 `gtk.main_quit()` 函数来关闭我们的应用程序。

5.4 以上就是把我们的代码和 GUI 中我们的 `window1` 的信号连接起来所需的全部操

作。

5.5 点击“Save（保存）”按钮。

5.6 现在，我们该看看我们的窗口变成什么样子了。

6 Python 运行 第二次

6.1 选择终端所在桌面，点选终端窗口。

6.2 输入下面的命令运行我们的应用程序。

```
python glade-tut.py
```

6.3 我们的窗口现在将会出现在桌面的中心。

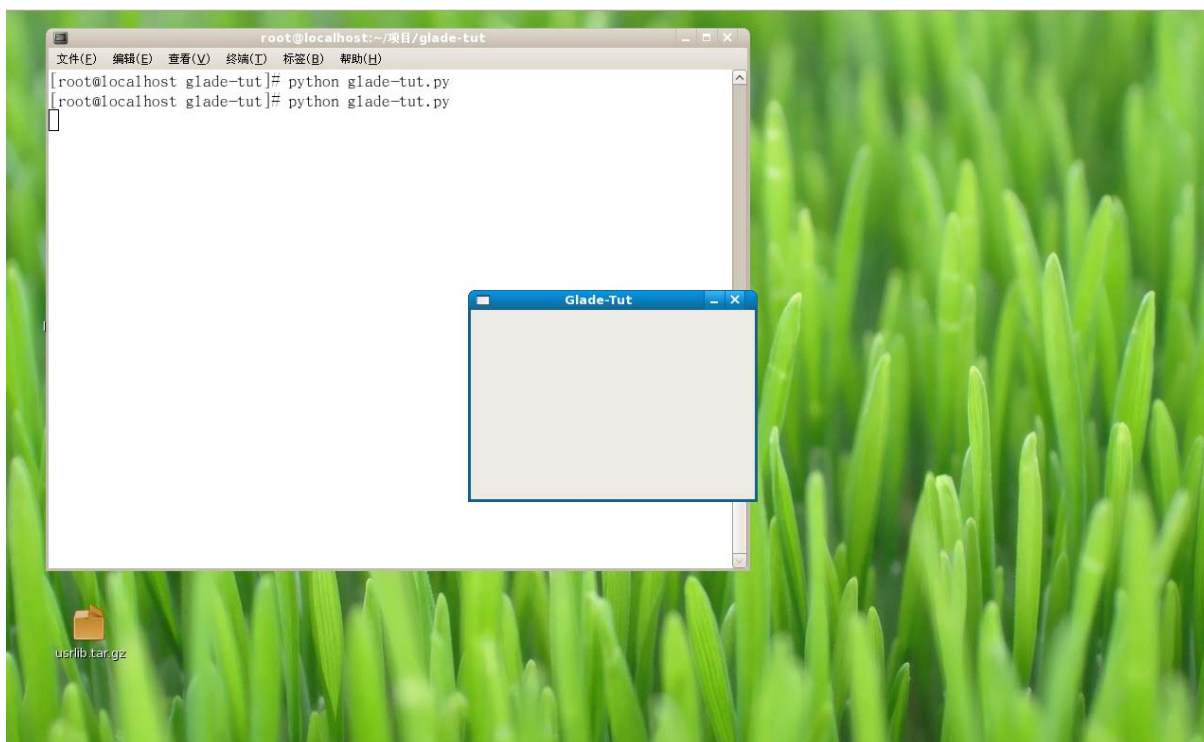


图 18 第二次运行 glade-tut.py

6.4 你将看到现在的窗口标题是 Glade-Tut，它的大小是我们设定好的并且不能改变大小。

6.5 关闭窗口我们看到它返回到了提示符状态，这是因为我们设置的 `destroy` 信号确保了我们的应用程序正确地关闭。

6.6 现在让我们一起为我们的应用程序添加一些其他的控件，比如按钮等等。

7 Glade 添加一个按钮

7.1 选择 Glade 所在桌面。

7.2 选择调色板窗口上的“GTK+ Basic”按钮 


7.3 点击“Vertical Box（垂直框）”按钮 ，然后点击 window1 窗口的任何地方，系统会弹出一个行数设置对话框。



图 19 行数设置对话框

7.4 不用修改，保留默认值 3，然后点击“OK（确定）”按钮。

7.5 一个垂直的容器将会出现在 window1 窗口中，它有三行。



图 20 vbox 垂直容器

7.6 Vertical Box（垂直框）的每一行都是一个容器，可以单独被选择。尝试点击每一行，看看它会有什么反应。

7.7 点击部件树（Widget Tree）窗口中 window1 条目左侧的小三角，展开后我们看到了 vbox1 （你的属性窗口显示的 vbox 的名字可能会和教程中不同）条目。



图 21 vbox 条目

7.8 点击 vbox1 条目，它的属性就会显示在属性窗口中。



图 22 vbox 属性

7.9 我们暂时先不改变它的任何属性，一旦我们在 vbox 中放置一个部件，那么它将更有意义。

7.10 选择调色板窗口上的“GTK+ Basic”按钮 .

7.11 点击“Button（按钮）”按钮，然后点击 window1 的 vbox 的最上面一行的任何地方。

7.12 你的 Glade 桌面现在应该变成了这样。

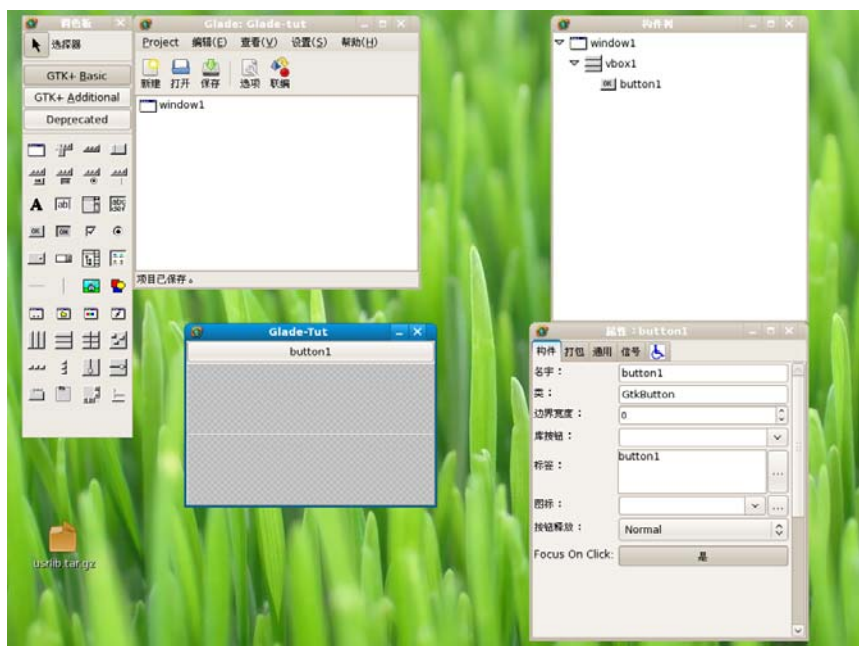


图 23 增加 button 后的 Glade 桌面

7.12.1 window1 的 vbox 的最上面一行有了一个新的按钮。

7.12.2 部件树（Widget Tree）窗口在列表中显示了新添加的按钮。

7.12.3 属性（Properties）窗口显示了这个新按钮的属性。

7.13 点击 Glade 主窗口的“Save（保存）”按钮。

7.14 在我们开始修改 vbox 容器和 button（按钮）部件的属性之前，让我们再次运行一下我们的应用程序，看看它现在是什么样子的。

8 Python 运行 第三次

8.1 选择终端所在桌面，点选终端窗口。

8.2 输入下面的命令运行我们的应用程序。

```
python glade-tut.py
```

8.3 我们的窗口现在如下图所示。



图 24 第三次运行 glade-tut.py

8.4 我们添加的按钮在顶端被压扁了，并且被拉伸成和整个窗口一样宽。

8.5 这个按钮在点击时会有按下的效果，但是不会触发任何事件，还需要我们为它设定信号才行。

8.6 现在我们返回到 Glade，去看看我们应该如何修改 vbox 的属性以及为我们的按钮添加信号。

9 Glade 配置按钮部件

9.1 选择 Glade 所在桌面。

9.2 点击我们放置在 window1 中的 button（按钮）部件，属性窗口将显示它的属性。

9.3 更改 Name（名称）为 btnNumberOne。

9.4 更改 Label（标签）为 Number One。

9.5 点击 Packing（打包）选项卡。

9.6 “Position（位置）”当前默认设置为 0，其含义是它现在放置在 vbox 容器的第一行。

9.6.1 使用键盘上的上下箭头键（或者文本框右侧的上下箭头）调整按钮在 vbox 容器中的所在行，这个值从 0 到 1 再到 2，观察一下这个值的变化和 button 在 vbox 容器中的位置的变化情况。

9.7 更改 Expand（扩展）的值为 Yes（是）。

9.8 更改 Fill（填充）的值为 Yes（是）。

9.9 现在，btnNumberOne 填满了 vbox 容器的整个第一行。

9.10 点击 Signals（信号）选项卡。

9.11 在 Signals（信号）选项卡中点击标记为“...”的按钮。

9.12 选择 GTKButton Signals（信号）类别下的“Clicked”，然后点击“OK（确定）”。

9.13 点击 Signals（信号）选项卡的“Add（添加）”按钮把这个信号添加到列表中。

9.14 点击 Widget Tree（部件树）窗口中的 vbox 条目，属性窗口将显示它的属性。

9.15 点击 Widget（构件）选项卡。

9.16 更改 Border Width（边界宽度）为 10，这个选项修改整个 vbox 容器的边界宽度。

9.17 更改 Spacing（间距）为 10，这个选项修改 vbox 容器相邻两行的间距。

9.18 窗体 window1 现在是这个样子的。



图 25 window1

9.19 隔开之后，现在的按钮在我们的窗口中就显得漂亮多了。

9.20 你的 vbox 的 Properties（属性）窗口应该如下图所示。



图 26 vbox1 属性窗口

9.21 你的你的 button（按钮）的 Properties（属性）窗口应该如下图所示。

9.21.1 Widget（构件）选项卡。



图 27 btnNumberOne 属性窗口构件选项卡

9.21.2 Packing（打包）选项卡。



图 28 btnNumberOne 属性窗口打包选项卡

9.21.3 Signals (信号) 选项卡。

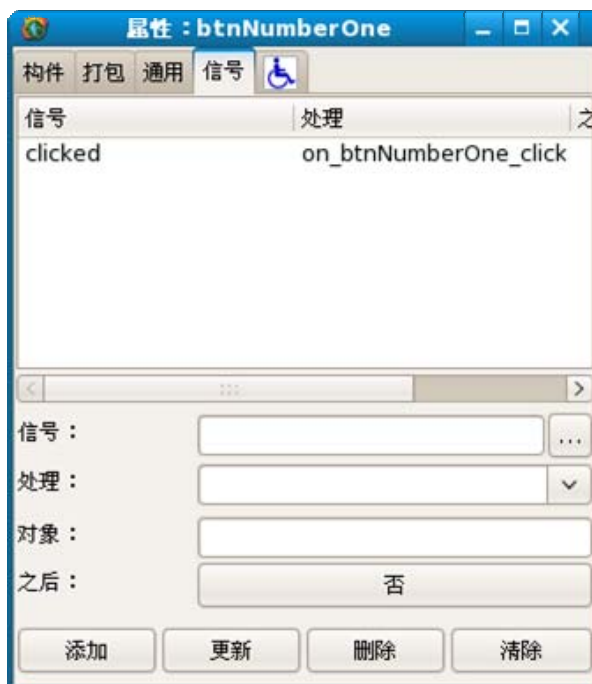


图 29 btnNumberOne 属性窗口信号选项卡

9.22 点击 Glade 主窗口上的“Save (保存)”按钮保存更改。

9.23 现在，让我们运行一下，看看我们的应用程序变成了什么样子。

10 Python 运行 第四次

10.1 选择终端所在桌面，点选终端窗口。

10.2 输入下面的命令运行我们的应用程序。

```
python glade-tut.py
```

10.3 我们的窗口现在如下图所示。



图 30 第四次运行 glade-tut.py

10.4 虽然没有完全达到我们想要的效果,但是我们可以看到我们的按钮现在已经有了一个美观的边缘。

10.5 现在,我们返回到 Glade 去完成 GUI 工作。

11 Glade 完成 GUI

11.1 我们现在将要完成 GUI 配置。现在我们将利用前面所学的基础知识来帮助我们完成它。

11.2 选择 Glade 所在的桌面。

11.3 选择调色板窗口上的“GTK+ Basic”按钮。



11.4 点击“Button (按钮)”按钮,然后点击 window1 的 vbox 的第二行的任何地方。

11.5 点击“Label (标签)”按钮,然后点击 window1 的 vbox 的第三行的任何地方。

11.6 窗体 window1 现在将如下图所示。



图 31 window1

11.7 我们现在需要调整新按钮和标签的属性使之更加美观。

- 11.8 点击刚刚添加的 **button**（按钮），在属性（**Properties**）窗口显示其属性
- 11.9 点选属性（**Properties**）窗口的 **Widget**（构件）选项卡。
- 11.10 更改 **Name**（名称）为 **btnNumberTwo**。
- 11.11 更改 **Label**（标签）为 **Number Two**。
- 11.12 点选 **Packing**（打包）选项卡。
- 11.13 更改 **Expand**（扩展）的值为 **Yes**（是）。
- 11.14 更改 **Fill**（填充）的值为 **Yes**（是）。
- 11.15 点击 **Signals**（信号）选项卡。
- 11.16 在 **Signals**（信号）选项卡中点击标记为“...”的按钮。
- 11.17 选择 **GTKButton Signals**（信号）类别下的“**Clicked**”，然后点击“**OK**（确定）”。
- 11.18 点击 **Signals**（信号）选项卡的“**Add**（添加）”按钮把这个信号添加到列表中。
- 11.19 点选属性（**Properties**）窗口的 **Widget**（构件）选项卡。
- 11.20 更改 **Name**（名称）为 **lblOutput**。
- 11.21 删除“**Label**（标签）”域的内容。
- 11.22 点击 **Glade** 主窗口上的“**Save**（保存）”按钮保存更改。
- 11.23 我们的 **GUI** 现在已经完成了,现在我们要做的就是添加最终代码前对应用程序进行一下测试。

12 Python 运行 第五次

- 12.1 选择终端所在桌面，点选终端窗口。
- 12.2 输入下面的命令运行我们的应用程序。

```
python glade-tut.py
```

- 12.3 我们的窗口现在如下图所示。



图 31 第五次运行 **glade-tut.py**

- 12.4 现在，我们有了两个漂亮的按钮。

12.5 在 window1 中放置的标签现在是看不见的，因为我们删除了它的内容。

12.6 现在我们要做的是添加一些代码来对这两个按钮的 signals（信号）和一些 callbacks（回调函数），当我们点击它们的时候在 lblOutput 上显示文本。

13 Gedit 完成 Python 代码

13.1 选择 Gedit 所在桌面，选中 Gedit 窗口。

13.2 修改 “__init__(self)” 函数如下所示。

```
#!/usr/bin/env python                                # when glade-tut.py is made executable run using python
import sys                                           # import sys library
try:
    import pygtk                                    # import pygtk library
    pygtk.require("2.0")                            # ensure we don't use earlier versions
except:
    pass
try:
    import gtk                                      # import gtk library
    import gtk.glade                               # import gtk.glade library
except:
    sys.exit(1)                                     # exit if import failed

class GladeTut:                                     # define our class

    def __init__(self):                             # initialisation function
        self.gladefile = "glade-tut.glade"          # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile)  # load the Glade XML from our file

        dic = {"on_btnNumberOne_clicked" : self.btnNumberOne_clicked,
                "on_btnNumberTwo_clicked" : self.btnNumberTwo_clicked,
                "on_window1_destroy" : gtk.main_quit } # create dictionary for connections
        self.wTree.signal_autoconnect(dic)          # connect our code to the signal

    def btnNumberOne_clicked(self, widget):          # callback for btnNumberOne
        lblRef=self.wTree.get_widget("lblOutput")
        lblRef.set_text("You clicked button Number One.")

    def btnNumberTwo_clicked(self, widget):          # callback for btnNumberTwo
        lblRef=self.wTree.get_widget("lblOutput")
        lblRef.set_text("You clicked button Number Two.")

if __name__ == "__main__":                          # if glade-tut.py is called rather than imported run this code
    frm = GladeTut()                                # create an instance of our class
    gtk.main()                                       # call GTK loop to run our application
```


13.3 我们修改 “dic=”，使字典中包括两个按钮回调函数的参考。

13.4 然后添加两个回调函数。

13.4.1 回调函数首先从字典中为 label（标签）lblOutput 获取一个参考。

13.4.2 然后利用这个参考设置 label（标签）的文本为 “You clicked button Number One.”。

13.4.3 两个按钮的回调函数的设计基本上一样。

13.5 至此，我们的应用程序已经完成了，点击 “Save（保存）” 按钮。

14 Python 运行 第六次

14.1 选择终端所在桌面，点选终端窗口。

14.2 输入下面的命令运行我们的应用程序。

14.3 现在，在我们的窗口中点击 Number One 按钮后如下图所示。



图 32 第六次运行 glade-tut.py——点击 Number One 按钮

14.4 点击 Number Two 按钮后如下图所示。



图 33 第六次运行 glade-tut.py——点击 Number Two 按钮

14.5 至此，你已经掌握了一个简单的 GUI 应用程序的基本设计方法。希望对在你今

后使用 Glade 进行开发时能够有所帮助。

14.6 你可以参考第 15 节，我给大家分享了一些有用的链接。

15 结束语

我希望这个教程已经使你理解了如何用 Glade、Gedit 和 Python 创建一个 GUI 的基本方法。

我希望你能够把你所学到的，当然还有对这篇教程的任何意见和建议反馈给我。我将利用这些信息来改进我的教程或者撰写其他有关 Linux 下创建 GUI 方面的教程。

这里我给大家一些有用的链接，可能在以后的学习中对你有所帮助：

<http://heather.cs.ucdavis.edu/~matloff/python.html>

<http://python-forum.org/py/index.php>

<http://www.pygtk.org/pygtk2reference/>

<http://glade.gnome.org/>

<http://www.python.org/>

Hope you enjoyed it!