PyGTK 2.0 Tutorial

John Finlay

October 25, 2012

Contents

Chapter 1

介绍

PyGTK 2.0 是一套提供GTK+ 2.X的python接口的Python模块. 在整个的其余部分文件PyGTK的是指PyGTK的2.
GTK +的2.X版本的GTK+。

主要网站PyGTK的www.pygtk.org [http://www.pygtk.org]。

PyGTK的主要作者:

James Henstrdge james@daa.comau[mailto:james@daa.com.au]

Python是一个可扩展的，面向对象的解释性编程语言，它提供 了一组丰富的模块，提供了大量的操作系
(如HTML，XML，FTP等),图形（包括OpenGL，TK等）,字符串 处理函数,邮件服务(IMAP，SMTP，POP3等),多
和加密服务。此外，还有许多其他的模块可从第三 当事人提供许多其他服务。 Python是授权根据类似的LG
的条款和供的Linux，UNIX，Windows和Macintosh操作系统. 关于Python的更多信息可在 www.python.org

Python的主要作者:

Gui van Rossum guido@python.org[mailto:guido@python.org]

GTK（GIMP工具包）是一个库，用于创建图形用户界面。这是授权 使用LGPL许可证，所以你可以开发开
是商业的非自由软件使用GTK，而不必花费任何牌照或特许权使用费。

这就是所谓的，因为它最初是为开发GNU图像处理程序GIMP工具包 (GIMP)，但GTK目前已被用于大量的软
模型环境（GNOME）项目。这基本上是一个GTK建立在GDK(GIMP绘图工具包)的 用于访问底层的窗口函数（X
Windows系统）。

GTK的主要作者是：

Peter Mattis petm@xcf.berkeley.edu [mailto:petm@xcf.berkeley.edu] Spencer Kimball spencer@xcf.berkeley.edu [mailto:spencer@xcf.berkeley.edu] Josh MacDonald jmacd@xcf.berkeley.edu [mailto:jmacd@xcf.berkeley.edu] GTK目前维护者： Owen Taylor otaylor@redhat.com [mailto:otaylor@redhat.com] Tim Janik timj@gtk.org [mailto:timj@gtk.org]

GTK本质上是一个面向对象的应用程序接口（API）。虽然完全用C编写的， 使用类和回调函数（函数指
还设有一个称为GLib的第三成分，包含了一些替代一些标准的呼叫，以及一些 处理链表的额外的功能，使
在这里实现的功能是不可用或其他的unix系统如g_strerror（）是非标准的。 有的还含有增强的libc版本
在2.0版本中，GLib的拿起了类型系统构成的基础GTK的类层次结构，信号 系统，用于整个GTK，一个线程A
平台和设备加载模块。

作为最后一个组件，GTK国际化文本输出，使用Pango库。 本教程描述了Python接口，GTK
+是基于GTK + 2.0教程写的托尼大风， 伊恩主。本教程试图尽可能多的PyGTK记录，但并不完整。

本教程假定Python的一些理解，以及如何创建和运行Python程序。如果你不 熟悉Python，请阅读的Pyt
[http://www.python.org/doc/current/tut/tut.html的]第一个。这 教程不承担GTK的理解，如果你正在学
你是如何找到本教程中，你遇到了麻烦。本教程并没有说明如何编译或安装 Python和GTK +或PyGTK的。
这份教程是基于: GTK+ 2.0 through GTK+ 2.4 Python 2.2 PyGTK 2.0 through PyGTK 2.4
这些示例是在RedHat 9.0系统，编写和测试。 该文件是一个“正在进行的工作”。敬请期待更新www.pygtk
我很希望听到你有学习PyGTK的从本文档中的任何问题，将不胜感激 如何可以提高输入。请贡献的详细信息
错误请对pygtk的项目，提交一个bug在bugzilla.gnome.org [http://bugzilla.gnome.org。信息
www.pygtk.org [http://www.pygtk.org/feedback.html有关Bugzilla的帮助。 PyGTK的2.0参考手册提供ht

PyGTK的类。 PyGTK的网站（www.pygtk.org [http://www.pygtk.org]）包含了其他有用的学习资源
PyGTK的包括广泛的常见问题[http://www.async.com.br/faq/pygtk/]和其他文章和教程的链接
和活跃的邮件列表和IRC频道（]的详细信息，请参阅www.pygtk.org [http://www.pygtk.org/feedback.html）

1.1 探索PyGTK

约翰·达林写了一个小的Python在Linux上运行的程序(pygtkconsole.py[的例子/pygtkconsole.py])
允许的PyGTK的互动探索。该方案提供了Python的交互式解释器接口，与一个子进程执行的，输入的命令。默认
会议是一个简单的例子：

```
moe: 96:1095$pygtkconsole.py
Python 2.2.2, PyGTK 1.99.14 (Gtk+ 2.0.6)
Interactive console to manipulate GTK+ widgets.
>>> w=Window()
>>> b=Button''(''Hello)
>>> w.add(b)
>>> def hello(b):
...
print "Hello, World!"
...
>>> b.connect''(''clicked, hello)
5
>>> w.show_all()
>>> Hello, World!
Hello, World!
Hello, World!
>>> b.set_label("Hi There")
>>>
```

这将创建一个窗口，包含一个按钮，点击时打印一条消息(“你好，世界！”)。 这个程序使得它轻易尝试
我也用一个程序，由Brian McErlean开发ActiveState的配方65109 [http://aspn.activestate.com/ASPN/C
使其运行PyGTK的2.X使用。我称它gpython.py的例子/ gpython.py]。 它的工作方式类似于pygtk-
console.py [的例子/ pygtkconsole.py]程序。

Note 这两个程序，不知道在Microsoft Windows，因为它们依赖于特定的Unix 接口。

Chapter 2

起步

首先我们介绍PyGTK的，我们先从最简单的程序可能。这个程序 (base.py[例子/ base.py的])将创建一个20
有没有办法退出，除了被杀死的使用壳。 You can run the above program using: python
base.py

如果base.py [的例子/ base.py]可执行文件，并且可以在你的PATH环境变量， 它可以运行使用：
base.py 第一行将在这种情况下，调用Python运行base.py [例子/ base.py的]。 第5-6行的帮助区分不同的
这些行指定我们想要使用PyGTK的版本的2.0，涵盖了所有版本的PyGTK的 主要2号。这防止了使用较早的程
系统上安装。 18-20行检查，如果__name__变量“__main__”， 表示正在运行的程序，直接从Python，而不
在这种情况下，程序将创建一个新的Base类的实例，并保存的参考 它在可变碱。然后，它调用的方法main
+的事件处理循环。

类似于图2.1，“简单的PyGTK的窗口”应该在您的显示器上弹出一个窗口。

nux或UNIX shell程序被调用的程序base.py [的例子/ base.py] 假设Python是你的PATH。这条线将所有
5-7行导入PyGTK的模块和初始化GTK +环境。 PyGTK的模块定义了蟒蛇 将在程序中使用的GTK
+功能的接口。对于那些熟悉GTK +的初始化 包括调用gtk_init（）函数。这几件事情对我们来说，如默认的
默认的信号处理程序，并检查参数传递给应用程序的命令行，寻找一个或 多个下列的： •
--gtk-module • --g-fatal-warnings • --gtk-debug • --gtk-no-debug • --gdk-debug •
--gdk-no-debug • --display • --sync • --name • --class

它从参数列表中删除这些留下任何它不承认你的应用程序进行解析或忽略。 这是一组接受所有GTK
+应用程序的标准参数。 第9-15行定义一个Python的名为Base的类，它定义了一个类的实例初始化方法
__init__（）。 在__init__（）函数创建一个顶层窗口（第11行），并指示显示GTK +（12号线）。
“ 在第11行的的参数gtk.WINDOW_TOPLEVEL，指定我们要在创建gtk.Window 窗口进行窗口管理器的装饰和有
没有孩子的默认设置为200×200，所以你仍然可以操作它。 第14-15行定义的main（）方法调用的PyGTK的
+主 事件处理循环处理鼠标和键盘事件以及窗口事件。 18-20行让程序自动启动，如果直接调用作为参数传
的解释，在这种情况下，程序的名称，在python变量__name__是字符串 “__main__”和18-20行中的代码的将
使用一个import语句，线条18-20将不会被执行。 第19行创建了一个称为基的基类的实例。一个gtk.Windo
第20行调用基类的main（）方法启动GTK +的事件处理循环。当控制 达到这一点，GTK +睡觉等待X事件（如
通知发生。然而，在我们这个简单的例子中，事件被忽略。

2.1 PyGTK的Hello, World

PyGTK的模块中定义的变量和函数命名为gtk的*。例如， helloworld.py [例子/ helloworld.py]程序所使用
False gtk.mainquit() gtk.Window() gtk.Button()

从PyGTK的模块。在以后的章节中，我将不指定GTK模块的前缀，但它会假设。 “ 示例程序将使用该模

2.2 信号和回调理论

Note

在GTK+2.0版本中，信号系统存GTK中移到了GLib中。我们不会深入探究GLib2.0 信号系统的扩展。这对于P

在深入探究helloworld.py之前[exmaples/helloworld.py]，我们将探讨一下信号与 回调。GTK+是一个事件

发生并且控制被传递到适当的函数。

这种控制传递使用信号来做。(这些信号与UNIX系统中的信号是不相同的，并且不是 使用它们实现的，虽然

当一个事件发生时，例如鼠标按钮点击， 被按下的widget将发出适当的信号。这就是GTK+如何处理它的大部分

所有widget都继承了的，例如"destroy"，也有widget特定的信号，例如toggle按钮上 的tog-
gled。

为了使一个按钮执行一个动作，我们设置信号处理函数并调用适当的函数。这是如下 使用Gtk-
Widget(from the GObject class)方法: handler_id = object.connect(name, func,
func_data) 其中object是用于发出信号的GtkWidget的实例，第一个参数name是一个包含要捕捉的
信号名称的字符串。第二个参数，func，是你想在信号捕捉之后被调用的函数，第三个 ，func_data,是你想传

或block的handler_id.

2.3 Events

除了上述的信号机制中，有一组X事件机制的事件的真实反映。 回调也可以连接到这些事件。这些事件包括：

为了连接一个回调函数的使用方法连接（这些事件），如上所述， 使用一个上述事件名称作为name参数。

不同形式的信号：

GdkEvent是一个Python的对象类型，其类型属性将显示上述事件已经发生。 “ 将取决于事件的类型的事件

NOTHING DELETE DESTROY EXPOSE SETTING

这些值将前面的事件类型gtk.gdk访问。例如gtk.gdk.DRAG_ENTER。 因此，连接一个回调函数，我们将使用

这假定按钮是一个GtkButton上的窗口小部件。现在，当鼠标在按钮和鼠标按钮 按下时，该函数but-
ton_press_callback将被调用。此功能可以被定义为：

从这个函数的返回值表示该事件是否应进一步传播的GTK +事件 处理机制。返回True表示该事件已经被处理

进一步。返回FALSE继续正常的事件处理。请参阅第20章，高级事件和信号处理 这个传播过程的详细信息。

GDK的选择和拖放和拖放的API也发出了一些事件中所反映的GTK +的信号。看 第22.3.2节，“在源窗口小部

get” 这些信号的回调函数的签名：

2.4 一步一步探究Hello World

现在，我们知道这背后的理论，让我们澄清走过的例子helloworld.py考试 - 普莱斯/ hel-
loworld.py程序。

第9-76行定义的HelloWorld类，它包含所有的回调对象的方法和对象的实例 初始化的方法。让我们来看看

第13-14行定义为hello（）回调方法，该按钮时，将被称为“点击”。当调用 方法，打印“你好世界”到

例如，但多数回调使用它们。数据定义PyGTK的，因为将使用默认值“无” 通过数据值，如果它不包括在con-
nect（）调用，这将引发一个错误，因为回调预期 三个参数，可能会收到只有两个。定义为“无”的默认值，

一直被称为只有两个参数（从来没有所谓的用户数据）。下面的例子将使用数据 参数告诉我们哪个按钮被按下

def hello(self, widget, data=None): print "Hello World"

下一个回调（16-26）有点特殊。窗口管理器的“delete_event”发生时，发送此事件 到应用程序。我们有

响应，或者干脆退出应用程序。

你这个回调函数返回的值让GTK +知道采取什么样的行动。通过返回TRUE，让它知道我们
不想发出“destroy”信号，保持我们的应用程序运行。返回FALSE，我们要求 “消灭”的发出，我们将依次调

stroy”信号处理程序。请注意评论已被移除 清晰度。

def delete_event(widget, event, data=None): print "delete event occurred"
return False

的destroy（）回调方法（行28-30）导致程序退出调用gtk.main_quit（）的。 函数告诉GTK +，它是退出从gtk.main（）时，控制返回到它。

第52行创建一个新按钮，节省了提到它在self.button。该按钮将标签为“Hello 世界“时显示。

self.button = gtk.Button("Hello World")

在第57行，我们的信号处理程序附加到按钮时，它会发出“clicked”信号，我们的hello（）回调方法被调用。如果我们没有为hello（），所以我们只是传递的数据没有任何数据。显然，“clicked”信号我们与我们的鼠标指针单击该按钮时发出。用户数据参数值无不是必需的。 可以除去。然后，回调将被称

self.button.connect("clicked", self.hello, None)

我们也将使用此按钮来退出程序。第62行说明了如何“消灭”的信号可能来自 无论是窗口管理器，或从回调，然后再下面的顺序设置了。你可能有你需要的尽可能多的回调， 和所有将要执行的顺序连接。

既然我们要使用的GtkWidget的destroy（）方法接受一个参数（窗口小部件被破坏 - 在这种情况下，窗口nect_object（）方法，并把它传递到窗口的参考。 “ connect_object（）方法安排通过的第一个回调参数的gtk.Widget destroy（）方法被调用时，它会导致从窗口发出“destroy”信号 这将反过来造成的的HelloWorld破坏（）方法被调用来结束程序。

第65行是一个包装的呼叫，将要说明在后面深度在第4章中，包装小部件。但它是相当容易理解。它只是告诉GTK +是被放置在窗口，它会显示该按钮。需要注意的是 GTK +容器只能包含一个部件。有多个部件以不同的方式。

self.window.add(self.button)

现在，我们拥有一切的方式，我们希望它是。与所有的信号处理程序中，按钮放置 在窗口中，它应该是+（66行和第69行）“显示在屏幕上的小部件。该窗口 窗口小部件，最后使整个窗口弹出一次，而不是看到它里面的按钮形成。虽然这样一个简单的例子，你永远也不会注意到。

self.button.show（） self.window.show（） 第73-75行定义的main（）方法，该方法调用的gtk.main

def main(self): gtk.main（）

第80-82行让程序自动运行，如果直接调用或参数的Python解释器。线 81创建的HelloWorld类的一个实HelloWorld类的ma in（）方法来启动GTK +的事件处理循环。

hello = HelloWorld() hello.main()

现在，当我们点击鼠标左键在GTK +按钮，窗口小部件会发出一个“clicked”信号。为了让大家使用这些信息，我们的程序设置了一个信号处理程序来捕获的信号，调度的功能，我们的选择。

在我们的例子中，我们创建按钮时，“点击”，hello（）方法被称为“无参数，并 然后该信号的下一个处stroy（）函数的窗口 作为它的参数，从而导致窗口发出“destroy”信号，它被捕获，并调用我们的Hello destroy（）方法

事件过程中使用的窗口管理器去关闭窗口，这将导致的“delete_event” 排放。这将要求我们的“del会发生什么。返回FALSE，将导致GTK +发出“destroy”信号导致的HelloWorld“消灭” 回调被调用，退出

Chapter 3

继续

```
int main(int argc, char* argv[]) {
        printf("Hello, world!\n");
        return 0;
}

int main(int argc, char* argv[]) {
        return 0;
}
```

3.1 More on Signal Handlers

让我们看一下connect()调用. object.connect(name, func, func_data) 一个connect()调用返回一个标识
每个信号和每个对象，因为你需要尽可能多的回调，都将依次执行的顺序，他们是 连接。
这个标签允许你通过从列表中删除此回调： object.disconnect(id) 因此，通过在标签返回的信号连接方
您也可以使用signal_handler_block() 和 signal_handler_unblock函数对暂时禁用 信号处理函数.

3.2 Hello World的升级

运行helloworld2.py examples/helloworld2.py产生在图3.1所示的窗口，“升级你好 世界例“。
Figure 3.1. Upgraded Hello World Example 有没有简单的方法来退出程序，你会发现这个时候，你必须使
排队要杀死它。的读者将是一个很好的锻炼插入第三个“退出”按钮，退出程序。您 也不妨玩的选项pack
并观察其行为。 上的第一个helloworld程序代码的不同之处是在一个简短的评论。 正如上面有没有“破坏
loworld的。 第13-14行定义一个回调方法，该方法是类似的hello（） 回调中的第一个helloworld。不同的
是的回调打印一条消息，包括数据传递的。 第27行设置一个标题字符串上使用的窗口的标题栏（参见图3.
Hello World例子，“升级”）。 第39行创建一个水平的的对话框（gtk.HBox）持有的两个按钮是建立在第
水平盒子的窗口容器。 49和64行的回调（） 方法来连接按钮的“clicked”信号。每个按钮都设置了不同的
字符串将被传递给回调函数（） 方法被调用时。 第53和66行包的水平框中的按钮。第57和70行要求GTK显示
线71-72问GTK的包装盒和窗口分别显示.

Chapter 4

打包小部件

Chapter 5

小部件概览

Chapter 6

按钮部件

Chapter 7

调整

Chapter 8

range部件

Chapter 9

杂项部件

Chapter 10

容器部件

Chapter 11

菜单部件

Chapter 12

绘画区域

Chapter 13

TextView部件

Chapter 14

Tree View部件

Chapter 15

PyGTK 2.2 的新部件

Chapter 16

PyGTK 2.4 的新部件

Chapter 17

Undocumented部件

Chapter 18

设置部件属性

Chapter 19

Timeouts,IO 和 Idle 函数

Chapter 20

高级事件和信号处理

Chapter 21

控制选择

Chapter 22

拖放(Drag and Drop)DND

Chapter 23

GTK的rc文件

Chapter 24

scribble，一个绘图程序范例

Chapter 25

写PyGTK程序的小贴士

Chapter 26

Contributing

Chapter 27

Credits

Chapter 28

教程版权和许可声明