第6章 调整对象

GTK有多种构件能够由用户通过鼠标或键盘进行调整，比如范围构件。还有一些构件，比如说GtkText和GtkViewport，内部都有一些可调整的属性。

很明显，当用户调整范围构件的值时，应用程序需要对值的变化进行响应。一种办法就是当构件的调整值发生变化时，让每个构件引发自己的信号 ,, 将新值传递到信号处理函数中，或者让它在构件的内部数据结构中查找构件的值。但是，也许需要将这个调整值同时连接到几个构件上，使得调整一个值时，其他的值也随之变化。最明显的例子就是将一个滚动条连接到一个视角构件或者滚动的文本区上。如果每个构件都有自己的设置或获取调整值的方法，程序员或许需要自己编写很复杂的信号处理函数，以便将这些不同构件之间的变化同步或相关联。

GTK用一个调整对象解决了这个问题。调整对象不是构件，但是为构件提供了一种以抽象、灵活的方法传递调整值信息的方法。调整对象最明显的用处就是为范围构件（比如滚动条和比例构件）储存配置参数和值。然而，因为调整对象是从 GtkObject派生的，在其正常的数据结构之外，它还具有一些特殊的功能。最重要的是，它们能够引发信号，就像构件一样，这些信号不仅能够让程序对用户在可调整构件上的输入进行响应，还能在可调整构件之间透明地传播调整值。

在许多其他的构件中都能够看到调整对象的用处。比如进度条、视角、滚动窗口等。

6.1 创建一个调整对象

许多使用调整对象的构件都能够自动创建它，但是有些情况下，必须自己手工创建。用下面的函数创建调整对象：

```
GtkObject *gtk_adjustment_new( gfloat value,
                               gfloat lower,
                               gfloat upper,
                               gfloat step_increment,
                               gfloat page_increment,
                               gfloat page_size );
```

其中的“ value ”参数是要赋给调整对象的初始值，通常对应于一个可调整构件的最高或最低位置。“ lower ”参数指定调整对象能取的最低值，“ step_increment ”参数指定用户能小步增加的值，“ page_increment ”是用户能大部调整的值。“ page_size ”参数通常用于分栏构件的可视区域。“ upper ”参数用于表示分栏构件的子构件的最底部或最右边的坐标。因而，它不一定总是“ value ”能取的最大值，因为这些构件的“ page_size ” 通常是非零值。

6.2 使用调整对象

可调整构件大致可以分为两组：一组对这些值使用特定的单位，另一组将这些值当作任意数值。后一组包括范围构件：滚动条、比例构件、进度条以及微调按钮。这些构件的值都

可以使用鼠标和键盘直接进行调整。它们将调整对象的 upper和lower值当作用户能够操纵的调整值的范围。缺省时，它们只会修改调整对象的值（也就是说，它们的范围一般都是不变的）。

另一组包含文本构件、视角构件以及滚动窗口构件。所有这些构件都是间接通过滚动条进行调整的。所有使用调整对象的构件都可以使用自己的调整对象，或者使用外部创建的调整对象，但是最好让这一类构件都使用它们自己的调整对象。

现在，你也许在想，文本构件和视角构件除了调整对象的值以外，其他的值都是由它控制的，而滚动条就是控制调整值的，如果在滚动条和文本构件之间共享调整对象，操纵滚动条会自动调整文本构件吗？确实如此，就像下面的代码所做的：

```
/* 创建自己的调整对象*/
text = gtk_text_new (NULL, NULL);

/*让垂直滚动条使用文本构件自己创建的调整对象 */
vscrollbar = gtk_vscrollbar_new (GTK_TEXT(text)->vadj);
```

6.3 调整对象内部机制

如果我想创建一个信号处理函数，当用户调整范围构件或微调按钮时让这个处理函数进行响应，应该从调整对象中取什么值，怎样从中取值呢？要解决这个问题，先看一眼_GtkAdjustment结构的定义：

```
struct _GtkAdjustment
{
  GtkData data;
  gfloat lower;
  gfloat upper;
  gfloat value;
  gfloat step_increment;
  gfloat page_increment;
  gfloat page_size;
};
```

应该知道的第一件事就是没有宏或存取函数能够从一个调整对象中取值。所以，必须自己完成这件工作。

因为设置调整对象的值时，通常想改变它的值以适应任何使用这个调整对象的构件，GTK提供了下面的函数：

```
void gtk_adjustment_set_value( GtkAdjustment *adjustment,
                               gfloat         value );
```

前面说过，和其他构件一样，调整对象是 GtkObject的子类，因而，它也能够引发信号。这也是为什么当滚动条和其他可调整构件共享调整对象时它们能够自动更新的原因。所有的可调整构件都为它们的调整对象的 value_changed信号设置了一个信号处理函数。下面是这个信号在_GtkAdjustmentClass结构中的定义：

```
void (* value_changed) (GtkAdjustment *adjustment);
```

各种使用调整对象的构件都会在它们的值发生变化时引发它们的调整对象的信号。这种情况发生在当用户输入值使范围构件的滑块移动和当程序使用 gtk_adjustment_set_value()函数显式地改变调整对象的值时。所以，如果有一个比例构件，想在它的值改变时改变一幅画的

旋转角度，应该创建像下面这样的回调函数：

```
void cb_rotate_picture (GtkAdjustment *adj, GtkWidget *picture)
{
  set_picture_rotation (picture, adj->value);
...
```

将这个回调函数连接到构件的调整对象上：

```
gtk_signal_connect (GTK_OBJECT (adj), "value_changed",
                    GTK_SIGNAL_FUNC (cb_rotate_picture), picture);
```

当构件重新配置了它的调整对象的upper和lower值时（比如，用户向文本构件添加了更多的文本时），发生了什么？在这种情况下，它会引发一个“ changed ”信号：

```
  void (* changed)       (GtkAdjustment *adjustment);
```

范围构件一般为这个信号设置回调函数，构件会改变它们的外观以反映变化。例如，滚动条上的滑块大小会根据它的调整对象的变化而变化。

一般不使用这个信号，除非想要写一个新的范围构件。不过，如果直接改变了调整对象的值，应该引发这个信号，以便重新配置它。用下面的函数引发这个信号：

```
gtk_signal_emit_by_name (GTK_OBJECT (adjustment), "changed");
```