第8章　范围构件 GtkRange

GtkRange(范围构件)是一大类构件，包含常见的滚动条构件和较少见的“比例”构件。尽管这两种构件是用于不同的目的，它们在功能和实现上都是非常相似的。所有范围构件共用一套公用的图形元素，每一个都有自己的 XWindow，接受自己的事件。它们都包含一个“滑槽”和一个“滑块”。用鼠标指针拖动滑块可以在滑槽中前后移动，在滑块前后的滑槽中点击，滑块就会前后大步移动。

和前面提到的调整对象一样，所有范围构件是与一个调整对象相关联的。该对象会计算滑块的长度和在滑槽中的位置。当用户操纵滑块时，范围构件会改变调整值。

8.1　滚动条构件GtkScrollBar

这是标准的滚动条。一般只用于一些需要滚动条的构件，比如列表、文本构件，或视角构件(在很多情况下使用滚动窗口构件更方便)。对其他目的,应该使用比例构件，因为它更友好，而且有更多的特性。

有水平和垂直滚动条两种类型。可以用下面的函数创建滚动条：

```
GtkWidget *gtk_hscrollbar_new( GtkAdjustment *adjustment);
GtkWidget *gtk_vscrollbar_new( GtkAdjustment *adjustment).
```

adjustment参数可以是一个指向已有调整对象的指针或 NULL，当为NULL时会自动创建一个。如果希望将新的调整值传递给其他构件，例如文本构件的构建函数，在这种情况下指定NULL是很有用的。

8.2　比例构件GtkScale

GtkScale(比例构件)一般用于允许用户在一个指定的取值范围内可视地选择和操纵一个值。例如，在图片的缩放预览中调整放大倍数，或控制一种颜色的亮度，或在指定屏幕保护启动之前不活动的时间间隔时，可能需要用到比例构件。

有两种不同类型的比例构件：水平的和垂直的比例构件。大多数程序员似乎喜欢水平的比例构件。既然在本质上它们的工作方法是相同的，那么不需要对它们分别对待。

用下面的函数创建水平和垂直的比例构件：

```
GtkWidget *gtk_vscale_new( GtkAdjustment *adjustment);
GtkWidget *gtk_hscale_new( GtkAdjustment *adjustment);
```

adjustment参数可以是一个已经用gtk_adjustment_new()创建的调整对象或NULL，此时,会创建一个匿名的调整对象，所有的的值都设为 0.0(在此处用处不大)。

为了避免引起困惑，可能要创建一个 page_size的值设为0.0的调整对象，让它的实际上限值与用户能选择的最高值相对应。

8.2.1　函数和信号

比例构件可以在滑槽的旁边以数字形式显示其当前值。默认行为是显示值 ,但是可以用下

面的函数改变其行为：

```
void gtk_scale_set_draw_value( GtkScale *scale, gint draw_value );
```

可以猜到，draw_value取值为TRUE或FALSE，结果是显示或不显示。

缺省情况下，比例构件显示的值，也就是在 GtkAdjustment定义中的value域，圆整到一位小数。可以用以下函数改变显示的小数位：

```
void gtk_scale_set_digits( GtkScale *scale, gint digits );
```

digits是要显示的小数位数。可以设置为任意位数，但是实际上屏幕上最多只能显示 13位小数。

最后，显示的值可以放在滑槽的不同位置：

```
void gtk_scale_set_value_pos( GtkScale *scale, GtkPositionType pos );
```

参数pos是GtkPositionType类型，<gtk/gtkenums.h>中有它的定义，可以取以下值之一：

```
GTK_POS_LEFT GTK_POS_RIGHT
GTK_POS_TOP GTK_POS_BOTTOM
```

如果将值显示在滑槽的“边上”(例如，在水平比例构件的滑槽的顶部和底部)，显示的值将跟随滑块上下移动。

8.2.2 常用的范围函数

范围构件本质上来说都是相当复杂的，不过，像所有“基本类”构件一样，绝大部分函数只有当你想彻底了解它们时才有用。另外,几乎所有函数和信号只在用它们写派生构件时才管用。但是，在<gtk/gtkrange.h>中还是有一些很有用的函数，并且在所有范围构件都起作用。

1. 设置更新方式

范围构件的“更新方式”定义了用户与构件交互时的调整值如何变化，以及调整值如何引发value_changed信号。更新方式在<gtk/gtkenums.h>中定义为GtkUpdateType枚举类型，有以下取值：

- GTK_UPDATE_POLICY_CONTINUOUS：这是默认值。value_changed信号是连续引发，例如，每当滑块移动，甚至移动最小数量时都会引发。
- GTK_UPDATE_POLICY_DISCONTINUOUS：只有滑块停止移动，用户释放鼠标键时才引发value_changed信号。
- GTK_UPDATE_POLICY_DELAYED：当用户释放鼠标键，或者滑块短期停止移动时才引发value_canged信号。

范围构件的更新方式可以用以下方法设置：用 GTK_RANGE(Widget)宏将变量转换为构件指针，并将它传递给以下函数：

```
void gtk_range_set_update_policy( GtkRange *range, GtkUpdateType policy) ;
```

2. 获得和设置调整值

可以用以下函数快速设置或取得调整值：

```
GtkAdjustment* gtk_range_get_adjustment( GtkRange *range );
void gtk_range_set_adjustment( GtkRange *range, GtkAdjustment *adjustment);
```

gtk_range_get_adjustment()：返回一个指向范围构件所连接的调整对象的指针。

gtk_range_set_adjustment()：如果将一个范围构件已经连接的调整传递到函数里面，什么也不会发生，不管是否改变其内部的值。

如果将新的调整对象传递进去，如果存在旧的调整对象，它会解除旧连接(可能会销毁它)，将适当的信号连接到新的调整对象，并且调用私有函数 gtk_range_adjustment_changed()，该函数将(或至少假装会)重新计算滑块的位置和尺寸，并重新绘出该构件。

正如在调整对象部分所提到的，如果想重新使用同一个调整对象，当直接修改它的值时，应该让它引发一个“ changed ”信号，如下所示：

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

8.2.3 键盘和鼠标绑定

所有的GtkRange在鼠标点击交互时方式的多少是相同的。在滑槽上单击鼠标左键使调整值加上或减去一个 page_increment值，滑块也移动相应的距离。在滑槽上点击鼠标右键将使滑块跳到鼠标点击处。在滚动条的箭头会使它的调整值改变一个 step_increment值。

要习惯使用它可能会花一点时间。不过，在 GTK中，缺省情况下的滚动条和比例构件可以获得键盘焦点。如果用户会感到困惑 ,可以在滚动条的以下函数中用 GTK_CAN_FOCUS标志禁用：

```
GTK_WIDGET_UNSET_FLAGS (scrollbar, GTK_CAN_FOCUS);
```

按键绑定(当然只在该构件获得焦点时有效)在水平和垂直范围构件上略有不同，理由很明显。但这些对比例构件和滚动条构件来说却不尽相同，原因也不太明显 (对滚动窗口中的水平和垂直滚动条来说，可以避免这种困惑，在这种情况下，它们都对同一个区域操作)。

1. 垂直范围构件

所有垂直范围构件能用向上和向下键操作，还可以用 PageUp 和PageDown操作。上下箭头使滑块以step_increment量上下移动， PageUp和PageDown使滑块以page_increment量上下移动。用户可以使用键盘让滑块在滑槽的两端之间自由移动。

对垂直比例构件，移动滑块是用 Home和End键，然而对垂直滚动条构件，用 Ctrl+PageUp和Ctrl+PageDown操纵滑块。

2. 水平范围构件

对水平范围构件，向左和向右键与垂直范围构件的向上和向下键起同样的作用。 Home和End键将滑块移动到滑槽的头部和尾部。

对水平比例构件，用 Ctrl+向左键以及 Ctrl+向右键使滑块以 page_increment量移动滑块，对水平滚动条构件，用Ctrl+Home和Ctrl+End做同样的工作。

8.2.4 示例

在一个窗口上放置了三个范围构件，都连接到同一个调整对象，并使用上面提到的一些调整范围构件参数的控制方法，这样可以看到这些构件的效果。

```
/* GtkRange示例开始rangewidgets.c */

#include <gtk/gtk.h>
GtkWidget *hscale, *vscale;
void cb_pos_menu_select( GtkWidget       *item,
                         GtkPositionType  pos )
{
    /* 设置比例构件的值*/
```

```
    gtk_scale_set_value_pos (GTK_SCALE (hscale), pos);
    gtk_scale_set_value_pos (GTK_SCALE (vscale), pos);
}

void cb_update_menu_select( GtkWidget     *item,
                            GtkUpdateType  policy )
{
    /* 设置比例构件的更新方式*/
    gtk_range_set_update_policy (GTK_RANGE (hscale), policy);
    gtk_range_set_update_policy (GTK_RANGE (vscale), policy);
}

void cb_digits_scale( GtkAdjustment *adj )
{
    /* 设置adj->value圆整的小数位数*/
    gtk_scale_set_digits (GTK_SCALE (hscale), (gint) adj->value);
    gtk_scale_set_digits (GTK_SCALE (vscale), (gint) adj->value);
}

void cb_page_size( GtkAdjustment *get,
                   GtkAdjustment *set )
{
/* 将示例调整对象的page_size和page_increment值设置
 * 为"Page Size指定的值*/
    set->page_size = get->value;
    set->page_increment = get->value;

/* 现在，引发一个"changed"信号，以重新配置所有
 * 已经连接到这个调整对象的构件*/
    gtk_signal_emit_by_name (GTK_OBJECT (set), "changed");
}

void cb_draw_value( GtkToggleButton *button )
{
    /* 根据开关按钮的状态设置在比例构件上是否显示比例值 */
    gtk_scale_set_draw_value (GTK_SCALE (hscale), button->active);
    gtk_scale_set_draw_value (GTK_SCALE (vscale), button->active);
}

GtkWidget *make_menu_item( gchar         *name,
                           GtkSignalFunc  callback,
                   gpointer       data )
{
    GtkWidget *item;

    item = gtk_menu_item_new_with_label (name);
    gtk_signal_connect (GTK_OBJECT (item), "activate",
                        callback, data);
    gtk_widget_show (item);
    return(item);
}
```

```
void scale_set_default_values( GtkScale *scale )
{
    gtk_range_set_update_policy (GTK_RANGE (scale),
                                 GTK_UPDATE_CONTINUOUS);

    gtk_scale_set_digits (scale, 1);
    gtk_scale_set_value_pos (scale, GTK_POS_TOP);
    gtk_scale_set_draw_value (scale, TRUE);
}

/* 创建示例窗口*/

void create_range_controls( void )
{
    GtkWidget *window;
    GtkWidget *box1, *box2, *box3;
    GtkWidget *button;
    GtkWidget *scrollbar;
    GtkWidget *separator;
    GtkWidget *opt, *menu, *item;
    GtkWidget *label;
    GtkWidget *scale;
    GtkObject *adj1, *adj2;

    /* 标准的创建窗口代码*/
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

gtk_signal_connect (GTK_OBJECT (window), "destroy",
                    GTK_SIGNAL_FUNC(gtk_main_quit),
                    NULL);
    gtk_window_set_title (GTK_WINDOW (window), "range controls");

    box1 = gtk_vbox_new (FALSE, 0);
    gtk_container_add (GTK_CONTAINER (window), box1);
    gtk_widget_show (box1);

    box2 = gtk_hbox_new (FALSE, 10);
    gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
    gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
    gtk_widget_show (box2);

/* 注意, page_size只对滚动条器件有区别, 并且, 实际上取得的最高值
 * 就是(upper - page_size)*/
    adj1 = gtk_adjustment_new (0.0, 0.0, 101.0, 0.1, 1.0, 1.0);

    vscale = gtk_vscale_new (GTK_ADJUSTMENT (adj1));
    scale_set_default_values (GTK_SCALE (vscale));
    gtk_box_pack_start (GTK_BOX (box2), vscale, TRUE, TRUE, 0);
    gtk_widget_show (vscale);

    box3 = gtk_vbox_new (FALSE, 10);
```

```
    gtk_box_pack_start (GTK_BOX (box2), box3, TRUE, TRUE, 0);
    gtk_widget_show (box3);

    /* 重新使用同一个调整对象*/
    hscale = gtk_hscale_new (GTK_ADJUSTMENT (adj1));
    gtk_widget_set_usize (GTK_WIDGET (hscale), 200, 30);
    scale_set_default_values (GTK_SCALE (hscale));
    gtk_box_pack_start (GTK_BOX (box3), hscale, TRUE, TRUE, 0);
    gtk_widget_show (hscale);

    /* 再次重用同一个调整对象*/
    scrollbar = gtk_hscrollbar_new (GTK_ADJUSTMENT (adj1));
    /* 注意，这导致当滚动条移动时，比例构件总是连续更新 */
    gtk_range_set_update_policy (GTK_RANGE (scrollbar),
                                 GTK_UPDATE_CONTINUOUS);
    gtk_box_pack_start (GTK_BOX (box3), scrollbar, TRUE, TRUE, 0);
    gtk_widget_show (scrollbar);

    box2 = gtk_hbox_new (FALSE, 10);
    gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
    gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
    gtk_widget_show (box2);

    /* 用一个检查按钮控制是否显示比例构件的值 */
    button = gtk_check_button_new_with_label("Display value on scale
widgets");
    gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button), TRUE);
    gtk_signal_connect (GTK_OBJECT (button), "toggled",
                        GTK_SIGNAL_FUNC(cb_draw_value), NULL);
    gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
    gtk_widget_show (button);

    box2 = gtk_hbox_new (FALSE, 10);
    gtk_container_set_border_width (GTK_CONTAINER (box2), 10);

    /* 用一个选项菜单以改变显示值的位置 */
    label = gtk_label_new ("Scale Value Position:");
    gtk_box_pack_start (GTK_BOX (box2), label, FALSE, FALSE, 0);
    gtk_widget_show (label);

    opt = gtk_option_menu_new();
    menu = gtk_menu_new();
    item = make_menu_item ("Top",
                           GTK_SIGNAL_FUNC(cb_pos_menu_select),
                           GINT_TO_POINTER (GTK_POS_TOP));
    gtk_menu_append (GTK_MENU (menu), item);
      item = make_menu_item ("Bottom", GTK_SIGNAL_FUNC (cb_pos_menu_select),
                           GINT_TO_POINTER (GTK_POS_BOTTOM));
    gtk_menu_append (GTK_MENU (menu), item);
    item = make_menu_item ("Left", GTK_SIGNAL_FUNC (cb_pos_menu_select),
                           GINT_TO_POINTER (GTK_POS_LEFT));
```

```
gtk_menu_append (GTK_MENU (menu), item);
item = make_menu_item ("Right", GTK_SIGNAL_FUNC (cb_pos_menu_select),
                       GINT_TO_POINTER (GTK_POS_RIGHT));
gtk_menu_append (GTK_MENU (menu), item);
gtk_option_menu_set_menu (GTK_OPTION_MENU (opt), menu);
gtk_box_pack_start (GTK_BOX (box2), opt, TRUE, TRUE, 0);
gtk_widget_show (opt);

gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
gtk_widget_show (box2);

box2 = gtk_hbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);

/* 另一个选项菜单，这里是用于设置比例构件的更新方式 */
label = gtk_label_new ("Scale Update Policy:");
gtk_box_pack_start (GTK_BOX (box2), label, FALSE, FALSE, 0);
gtk_widget_show (label);
opt = gtk_option_menu_new();
menu = gtk_menu_new();

item = make_menu_item ("Continuous",
                       GTK_SIGNAL_FUNC (cb_update_menu_select),
                       GINT_TO_POINTER (GTK_UPDATE_CONTINUOUS));
gtk_menu_append (GTK_MENU (menu), item);

item = make_menu_item ("Discontinuous",
                       GTK_SIGNAL_FUNC (cb_update_menu_select),
                       GINT_TO_POINTER (GTK_UPDATE_DISCONTINUOUS));
gtk_menu_append (GTK_MENU (menu), item);
item = make_menu_item ("Delayed",
                       GTK_SIGNAL_FUNC (cb_update_menu_select),
                       GINT_TO_POINTER (GTK_UPDATE_DELAYED));
gtk_menu_append (GTK_MENU (menu), item);

gtk_option_menu_set_menu (GTK_OPTION_MENU (opt), menu);
gtk_box_pack_start (GTK_BOX (box2), opt, TRUE, TRUE, 0);
gtk_widget_show (opt);

gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
gtk_widget_show (box2);

box2 = gtk_hbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);

/* 一个GtkHScale构件，用于调整示例比例构件的显示小数位数 */
label = gtk_label_new ("Scale Digits:");
gtk_box_pack_start (GTK_BOX (box2), label, FALSE, FALSE, 0);
gtk_widget_show (label);
adj2 = gtk_adjustment_new (1.0, 0.0, 5.0, 1.0, 1.0, 0.0);
gtk_signal_connect (GTK_OBJECT (adj2), "value_changed",
```

```
                    GTK_SIGNAL_FUNC (cb_digits_scale), NULL);
    scale = gtk_hscale_new (GTK_ADJUSTMENT (adj2));
    gtk_scale_set_digits (GTK_SCALE (scale), 0);
    gtk_box_pack_start (GTK_BOX (box2), scale, TRUE, TRUE, 0);
    gtk_widget_show (scale);
    gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
    gtk_widget_show (box2);

    box2 = gtk_hbox_new (FALSE, 10);
    gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
   /* 最后一个水平比例构件用于调整滚动条的page_size值 */
    label = gtk_label_new ("Scrollbar Page Size:");
    gtk_box_pack_start (GTK_BOX (box2), label, FALSE, FALSE, 0);
    gtk_widget_show (label);
    adj2 = gtk_adjustment_new (1.0, 1.0, 101.0, 1.0, 1.0, 0.0);
    gtk_signal_connect (GTK_OBJECT (adj2), "value_changed",
                    GTK_SIGNAL_FUNC (cb_page_size), adj1);
    scale = gtk_hscale_new (GTK_ADJUSTMENT (adj2));
    gtk_scale_set_digits (GTK_SCALE (scale), 0);
    gtk_box_pack_start (GTK_BOX (box2), scale, TRUE, TRUE, 0);
    gtk_widget_show (scale);
    gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
    gtk_widget_show (box2);
    separator = gtk_hseparator_new ();
    gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 0);
    gtk_widget_show (separator);

    box2 = gtk_vbox_new (FALSE, 10);
    gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
    gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, TRUE, 0);

    gtk_widget_show (box2);
    button = gtk_button_new_with_label ("Quit");
    gtk_signal_connect_object (GTK_OBJECT (button), "clicked",
                           GTK_SIGNAL_FUNC(gtk_main_quit),
                           NULL);
    gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
    GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
    gtk_widget_grab_default (button);
    gtk_widget_show (button);
    gtk_widget_show (window);
}
int main( int   argc,
          char *argv[] )
{
gtk_init(&argc, &argv);
create_range_controls();
gtk_main();
return(0);
}
/*示例结束 */
```

可以注意到程序没有对delete_event事件调用gtk_signal_connect函数，仅仅对destroy信号调用了该函数。但是destroy函数一样会执行，因为对给定窗口来说，未经处理的 delete_event事件会引发一个destroy信号。

将上面的示例代码保存为rangewidget.c，然后写一个如下所示的Makefile文件：

```
CC = gcc
rangewidgets: rangewidgets.c
   $(CC) `gtk-config --cflags`  rangewidgets.c -o \
         rangewidgets `gtk-config --libs`

clean:
   rm -f *.o rangewidgets
```

编译后，在shell提示符下输入./rangewidget执行该程序，运行效果见图8-1。

可以用鼠标指针调整对象的取值。点击Display value on scale widget检查按钮可以决定是否显示比例构件的值；在选项菜单Scale Value Position中选择合适的选项可以设置在何处显示比例构件的值；在选项菜单 Scale Update Policy中选择合适的值用来设置比例菜单的更新方式；滑动 Scale Digits可以设置比例构件的显示值的小数位数；滑动Scollbar Page Size设置滚动条构件的page_size值。点击Quit按钮退出应用程序。

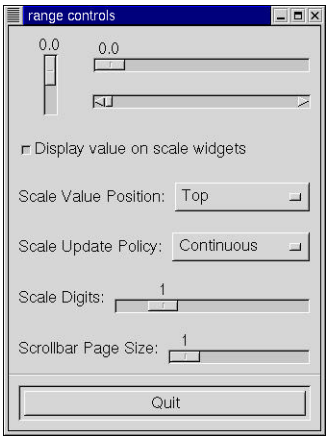

图8-1 范围构件