第7章　文本构件 GtkText

GtkText（文本构件）允许多行显示或编辑文本。它支持多种颜色以及多种字体的文本，允许它们以任何需要的形式混合显示，还有许多与 Emacs兼容的文本编辑命令。

文本构件支持完全的剪切/粘贴功能，还包括双击选择一个单词和三击选择整行的功能。

注意，请将GtkText和GtkEntry构件区分开。GtkEntry只能显示或编辑一行字符串，而不能将多种字体和多种颜色的文本混排。

7.1　创建、配置文本构件

创建新Text构件只有一个函数：

```
GtkWidget *gtk_text_new( GtkAdjustment *hadj,
                         GtkAdjustment *vadj );
```

其中的参数允许为文本构件指定水平和垂直的调整对象，并且可以用于跟踪构件的位置。向gtk_text_new函数传递NULL，函数会为文本构件创建自己的调整对象。

```
void gtk_text_set_adjustments( GtkText       *text,
                               GtkAdjustment *hadj,
                               GtkAdjustment *vadj );
```

上面的函数可以随时改变文本构件的水平和垂直的调整对象。

当文本构件中的文本超过构件能显示的空间时，文本构件不会自动显示滚动条。所以我们必须另行创建滚动条，将它们添加到要显示的窗口布局上。

```
vscrollbar = gtk_vscrollbar_new (GTK_TEXT(text)->vadj);
gtk_box_pack_start(GTK_BOX(hbox), vscrollbar, FALSE, FALSE, 0);
gtk_widget_show (vscrollbar);
```

上面的小段代码创建了一个垂直滚动条，并将它添加到文本构件的垂直 adjustment构件上，然后将它们组装到一个“组装盒”中。但是文本构件目前不支持水平滚动条。

文本构件有两个主要用途：允许用户编辑一段文本，或向用户显示多行文本。为了在两种操作模式之间进行切换，文本构件有以下函数：

```
void gtk_text_set_editable( GtkText *text,
                            gint     editable );
```

其中，editable参数可以是TRUE或FALSE，它指定用户是否可以编辑文本内容。当 Text构件是可编辑的时，会在当前插入点显示一个光标。

当然，不仅可以使用文本构件的这两种模式。还可以随时切换构件的可编辑模式，随时插入文本。

文本构件在文本如果太长，一行显示不下时会换行。缺省方式是在单词之间分行，可以用以下函数将其改变：

```
void gtk_text_set_word_wrap( GtkText *text,
                             gint     word_wrap );
```

这个函数允许我们指定文本构件是否在单词之间换行。 word_wrap参数的值可以是TRUE

或FALSE。

7.2 操作文本

可以用以下函数设置文本构件的插入点：

```
void gtk_text_set_point( GtkText *text,
                         guint    index );
```

index参数是要设置插入点的位置。

与上面的函数类似，使用下面的函数可以获得当前的插入点：

```
guint gtk_text_get_point( GtkText *text );
```

下面的函数可以与上面的函数联合应用：

```
guint gtk_text_get_length( GtkText *text );
```

返回当前文本的长度。长度是整个文本的字符数，其中还包括换行符等。

为了在当前插入点插入文本，可以使用 gtk_text_insert函数。插入时可以指定文本的背景色、前景色和字体。

```
void gtk_text_insert( GtkText    *text,
                      GdkFont    *font,
                      GdkColor   *fore,
                      GdkColor   *back,
                      const char *chars,
                      gint        length );
```

向fore、back、font中传递NULL参数让插入的文本使用构件内部的颜色和字体设置。设置length参数为-1，将字符串全部插入。

文本构件是一种动态重绘自身的构件 ，它会在gtk_main()函数之外重绘构件。这意味着文本构件内的所有变化都会立即生效。如果文本构件内的变化很多时，可能会引起闪烁。要在文本构件内的文本变化较大时不让构件重绘，可以先“ 冻结 ”构件，临时停止动态重绘本身。构件内的更新结束时，再将构件“ 解冻 ”。

下面两个函数“ 冻结 ”、“ 解冻 ’ 文本构件：

```
void gtk_text_freeze( GtkText *text );
void gtk_text_thaw( GtkText *text );
```

用以下函数删除当前位置以前或以后的 nchars个字符。返回值TRUE或FALSE指明操作成功或失败。

```
gint gtk_text_backward_delete( GtkText *text,
                               guint    nchars );
gint gtk_text_forward_delete ( GtkText *text,
                               guint    nchars );
```

如果要从文本构件取得文本的内容， GTK_TEXT_INDEX(t, index)宏可以取得t构件指定位置index处的字符。

用下面的函数取得大段文本：

```
gchar *gtk_editable_get_chars( GtkEditable *editable,
                               gint         start_pos,
                               gint         end_pos );
```

这实际上是文本构件的父类的函数。 end_pos设为-1指明文本的尾部。注意，索引值是从 0

开始的。

这个函数为文本分配一段内存，所以使用结束后别忘了用 g_free释放内存。

7.3 键盘快捷键

文本构件有许多预设的键盘快捷键，可用于常用的编辑、移动和选择等功能。它们常用Ctrl和Alt与其他键的组合键。

另外，按住Ctrl键的同时按方向键会让光标以单词为单位移动，而不是一个个地移动字符。按住Shift然后按方向键会选中或取消选择。

1. 移动快捷键

Ctrl-A 移到行头
Ctrl-E 移到行尾
Ctrl-N 移到下一行
Ctrl-P 移到前一行
Ctrl-B 向后一个字符
Ctrl-F 向前一个字符
Alt-B 向后一个单词
Alt-F 向前一个单词

2. 编辑快捷键

Ctrl-H 删除前一个字符(退格键)
Ctrl-D 删除下一个字符(删除键)
Ctrl-W 删除后一个单词
Alt-D 删除下一个单词
Ctrl-K 删除到行尾
Ctrl-U 删除一行

3. 选择快捷键

Ctrl-X 剪切到剪贴板
Ctrl-C 复制到剪贴板
Ctrl-V 从剪贴板粘贴

7.4 GtkText示例

```
/* 文本构件示例text.c */
/* text.c */
#include <stdio.h>
#include <gtk/gtk.h>

void text_toggle_editable (GtkWidget *checkbutton,
                           GtkWidget *text)
{
  gtk_text_set_editable(GTK_TEXT(text),
                        GTK_TOGGLE_BUTTON(checkbutton)->active);
}
```

```
void text_toggle_word_wrap (GtkWidget *checkbutton,
                            GtkWidget *text)
{
  gtk_text_set_word_wrap(GTK_TEXT(text),
                         GTK_TOGGLE_BUTTON(checkbutton)->active);
}

void close_application( GtkWidget *widget, gpointer data )
{
       gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *box1;
  GtkWidget *box2;
  GtkWidget *hbox;
  GtkWidget *button;
  GtkWidget *check;
  GtkWidget *separator;
  GtkWidget *table;
  GtkWidget *vscrollbar;
  GtkWidget *text;
  GdkColormap *cmap;
  GdkColor color;
  GdkFont *fixed_font;

  FILE *infile;

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_widget_set_usize (window, 600, 500);
  gtk_window_set_policy (GTK_WINDOW(window), TRUE, TRUE, FALSE);
  gtk_signal_connect (GTK_OBJECT (window), "destroy",
                      GTK_SIGNAL_FUNC(close_application),
                      NULL);
  gtk_window_set_title (GTK_WINDOW (window), "Text Widget Example");
  gtk_container_set_border_width (GTK_CONTAINER (window), 0);
  box1 = gtk_vbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (window), box1);
  gtk_widget_show (box1);

  box2 = gtk_vbox_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);

  table = gtk_table_new (2, 2, FALSE);
  gtk_table_set_row_spacing (GTK_TABLE (table), 0, 2);
```

```
gtk_table_set_col_spacing (GTK_TABLE (table), 0, 2);
gtk_box_pack_start (GTK_BOX (box2), table, TRUE, TRUE, 0);
gtk_widget_show (table);

/*创建GtkText构件*/
text = gtk_text_new (NULL, NULL);
gtk_text_set_editable (GTK_TEXT (text), TRUE);
gtk_table_attach (GTK_TABLE (table), text, 0, 1, 0, 1,
                  GTK_EXPAND | GTK_SHRINK | GTK_FILL,
                  GTK_EXPAND | GTK_SHRINK | GTK_FILL, 0, 0);
gtk_widget_show (text);

/*给GtkText构件添加垂直滚动条*/
vscrollbar = gtk_vscrollbar_new (GTK_TEXT (text)->vadj);
gtk_table_attach (GTK_TABLE (table), vscrollbar, 1, 2, 0, 1,
                  GTK_FILL, GTK_EXPAND | GTK_SHRINK | GTK_FILL, 0, 0);
gtk_widget_show (vscrollbar);

/* 取得系统颜色映射，将映射设置为红色 */
cmap = gdk_colormap_get_system();
color.red = 0xffff;
color.green = 0;
color.blue = 0;
if (!gdk_color_alloc(cmap, &color)) {
  g_error("couldn't allocate color");
}

/* 加载固定字体*/
fixed_font = gdk_font_load ("-misc-fixed-medium-r-*-*-*-140-*-*-*-*-*-*");

/* 实现文本构件
 * 可以插入一些文本了*/
gtk_widget_realize (text);

/*冻结text构件，准备多行更新*/
gtk_text_freeze (GTK_TEXT (text));

/* Insert some colored text */
gtk_text_insert (GTK_TEXT (text), NULL, &text->style->black, NULL,
                 "Supports ", -1);
gtk_text_insert (GTK_TEXT (text), NULL, &color, NULL,
                 "colored ", -1);
gtk_text_insert (GTK_TEXT (text), NULL, &text->style->black, NULL,
                 "text and different ", -1);
gtk_text_insert (GTK_TEXT (text), fixed_font, &text->style->black, NULL,
                 "fonts\n\n", -1);

/* 将text.c文件加载到text窗口*/

infile = fopen("text.c", "r");
if (infile) {
```

```
  char buffer[1024];
  int nchars;

  while (1)
    {
      nchars = fread(buffer, 1, 1024, infile);
      gtk_text_insert (GTK_TEXT (text), fixed_font, NULL,
                       NULL, buffer, nchars);

      if (nchars < 1024)
        break;
    }

  fclose (infile);
}

/* 将text构件"解冻", 让变化显示出来*/
gtk_text_thaw (GTK_TEXT (text));

hbox = gtk_hbutton_box_new ();
gtk_box_pack_start (GTK_BOX (box2), hbox, FALSE, FALSE, 0);
gtk_widget_show (hbox);

check = gtk_check_button_new_with_label("Editable");
gtk_box_pack_start (GTK_BOX (hbox), check, FALSE, FALSE, 0);
gtk_signal_connect (GTK_OBJECT(check), "toggled",
                    GTK_SIGNAL_FUNC(text_toggle_editable), text);
gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(check), TRUE);
gtk_widget_show (check);
check = gtk_check_button_new_with_label("Wrap Words");
gtk_box_pack_start (GTK_BOX (hbox), check, FALSE, TRUE, 0);
gtk_signal_connect (GTK_OBJECT(check), "toggled",
                    GTK_SIGNAL_FUNC(text_toggle_word_wrap), text);
gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(check), FALSE);
gtk_widget_show (check);

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 ("close");
gtk_signal_connect (GTK_OBJECT (button), "clicked",
                    GTK_SIGNAL_FUNC(close_application),
                    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);

    gtk_main ();
    return(0);
}
/*示例结束*/
```

将上述代码保存为text.c，然后写一段向下面这样的Makefile文件：

```
CC = gcc
text: text.c
   $(CC) `gtk-config --cflags`  text.c -o text `gtk-config --libs`
clean:
   rm -f *.o text
```

编译后，执行结果如图7-1所示。Editable检查按钮在按下时，文本构件内是可编辑的，如果该按钮是弹起的，文本构件内的文本是只读的。选中 Word Wrap检查按钮，文本会自动换行。

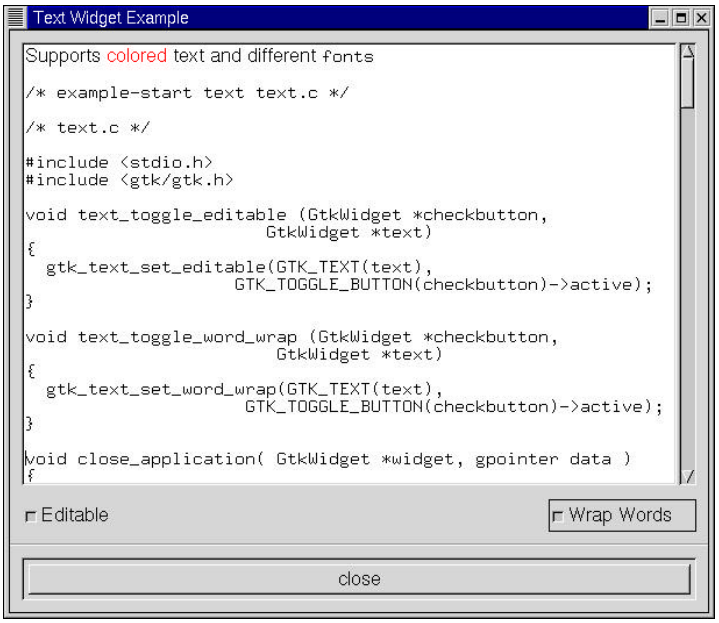

图7-1 文本构件示例