第11章　分栏列表构件GtkCList

GtkCList(分栏列表构件)是GtkList(列表构件)的替代品，但它提供更多的特性。分栏列表构件是多列列表构件，它有能力处理数千行的信息。每一列都可以有一个标题，而且可以是活动的。你还可以将函数绑定到列选择上。

11.1　创建分栏列表构件GtkCList

创建GtkClist构件的方法和创建其他构件的方法是类似的。有两种方法，一种是容易的，一种是难的。因为GtkCList可以有多列，因而在创建它之前，必须确定要创建的列表的列数。

```
GtkWidget *gtk_clist_new ( gint columns );
GtkWidget *gtk_clist_new_with_titles( gint   columns,
                                      gchar *titles[] );
```

第一种方式很简单，而第二种需要作一些解释。每一列都可以有一个与之相联系的标题，标题可以是一个标签构件，或者是一个按钮，只要能够对我们的动作作出响应。如果要使用第二种方式，则必须提供一个指向标题文本的指针，指针数目应该与列数相等。当然，我们可以用第一种方式，然后再手工添加标题以达到相同的目的。

注意，分栏列表构件没有自己的滚动条，如果要提供滚动条功能，应该将分栏列表构件放在一个滚动窗口构件中。

11.2　操作模式

有几个可以用于改变分栏列表构件行为的属性。先看下面这个：

```
void gtk_clist_set_selection_mode( GtkCList         *clist,
                                   GtkSelectionMode  mode );
```

就像函数名所暗示的一样，它设置了分栏列表的选择模式。第一个参数是要设置的分栏列表构件，第二个参数是单元的选择模式 (取值在gtkenums.h中有定义)。目前，有下面这些模式可以使用：

- GTK_SELECTION_SINGLE：选定内容为NULL，或包含一个指向单个被选中项目的Glist指针。
- GTK_SELECTION_BROWSE：如果GtkCList中不包含构件，或只包含不敏感的构件，则选定内容为NULL。否则，它包含一个指向 GList结构的GList指针，因而包含一个列表项。
- GTK_SELECTION_MULTIPLE：如果没有列表项被选中，选定内容为 NULL ；或者选定内容是一个指向第一个被选中列表项的指针。然后依次指向 GList结构中第二个被选中的列表项，等等。对GtkCList来说这是缺省模式。
- GTK_SELECTION_EXTENDED：选中内容总是NULL。

在GTK今后的版本中可能会增加其他模式。

还可以定义分栏列表构件的边框。使用以下函数完成定义：

```
void gtk_clist_set_shadow_type( GtkCList       *clist,
                                GtkShadowType  border );
```

第二个参数可能的取值是：

GTK_SHADOW_NONE

GTK_SHADOW_IN

GTK_SHADOW_OUT

GTK_SHADOW_ETCHED_IN

GTK_SHADOW_ETCHED_OUT

11.3 操作分栏列表构件列标题

创建分栏列表构件时自动创建相应的标题按钮。标题一般处于分栏列表构件窗口的顶部，它可以是能对鼠标点击响应的普通按钮，也可以仅仅是不会作任何响应的标签。有四个不同的函数调用帮助我们设置标题按钮的状态。

```
void gtk_clist_column_title_active( GtkCList *clist,
                                    gint     column );
void gtk_clist_column_title_passive( GtkCList *clist,
                                     gint      column );
void gtk_clist_column_titles_active( GtkCList *clist );
void gtk_clist_column_titles_passive( GtkCList *clist );
```

活动标题就是可以对用户动作响应的按钮标题，被动标题仅仅是一个标签。前两个函数激活或停用指定列的标题按钮，后两个激活或禁用整个分栏列表构件的按钮标题。

当然，有时候我们根本就不想使用标题按钮，那么可以用下面的函数显示或隐藏起来：

```
void gtk_clist_column_titles_show( GtkCList *clist );
void gtk_clist_column_titles_hide( GtkCList *clist );
```

有些情况下，标题对我们非常有用，需要有办法设置或改变它们。可以用以下函数来完成：

```
void gtk_clist_set_column_title( GtkCList *clist,
                                 gint      column,
                                 gchar     *title );
```

注意，一次只能设置一列的标题。如果知道标题应该是什么，应该在开始时用前面介绍的gtk_clist_new_with_titles函数创建分栏列表并设置标题。这样可以节省编写代码的时间，并且让程序更小。当然也有一些情况下手工设置这些值可能更好。有时候不是所有的标题都是文本。GtkCList构件为我们提供的标题按钮实际上能够和所有的构件结合起来使用，例如，它可以和pixmap构件结合起来，在上面显示一幅图片。使用下面得用函数可以为标题按钮设置构件：

```
void gtk_clist_set_column_widget( GtkCList  *clist,
                                  gint       column,
                                  GtkWidget *widget );
```

11.4 操纵列表

可以用以下的函数设置一列的对齐方式：

```
void gtk_clist_set_column_justification( GtkCList         *clist,
```

```
                                    gint                column,
                                GtkJustification  justification );
```

GtkJustification参数类型可取以下值：

- GTK_JUSTIFY_LEFT：列中的文本左对齐。
- GTK_JUSTIFY_RIGHT：列中的文本右对齐。
- GTK_JUSTIFY_CENTER：列中的文本居中对齐。
- GTK_JUSTIFY_FILL：文本使用列中所有可用的空间。它通常会在单词间插入额外的空格(如果是一个单词，会在单个字母间加空格)。许多“ 所见即所得 ”的文本编辑器具有这种特性。

下面的函数非常重要，在设置 GtkCList构件时应该作为标准加以使用。创建构件时各列的宽度是依据它们的标题确定的，因为多数情况下这不一定正确，用下面的函数设置列宽度：

```
void gtk_clist_set_column_width( GtkCList *clist,
                                 gint      column,
                                 gint      width );
```

注意，宽度是以像素度量，而不是以字母度量的。这对某列的单元格的高度也是同样的，但是缺省值是当前字体的高度，这对应用程序来说并不是至关重要的。用下面的函数设置行高度：

```
void gtk_clist_set_row_height( GtkCList *clist,
                               gint      height );
```

再次强调，高度也是以像素度量的。

可以在没有用户干预的情况下在列表之间随意移动，不过，我们需要能够直接跳到所需要的行和列，并且这个单元格应该是可见的。换句话说，就是我们应该能滚动列表，让单元格直接出现在可见的位置上。下面这个函数实现了这一点：

```
void gtk_clist_moveto( GtkCList *clist,
                       gint      row,
                       gint      column,
                       gfloat    row_align,
                       gfloat    col_align );
```

其中，row_align参数很重要。它是一个介于0.0和1.0之间的值，0.0意味着应该滚动列表，让所在行出现在顶部，如果 row_align的值是1.0，该行会出现在底部。所有介于 0.0到1.0之间的其他值表示位于顶部和底部之间的行。最后一个参数 col_align和row_align的作用一样，不过0.0是指左边，1.0是指右边。

根据应用程序的需要，我们不需要滚动那些我们已经能够看见的单元格。所以，怎样才能知道单元格是不是可见的呢？ 有一个函数能够做到这一点：

```
GtkVisibility gtk_clist_row_is_visible( GtkCList *clist,
                                        gint      row );
```

返回值可能是以下几个值之一：

- GTK_VISIBILITY_NONE 不可见
- GTK_VISIBILITY_PARTIAL 部分可见
- GTK_VISIBILITY_FULL 全部可见

注意，现在只能知道某一行是否可见，还没有办法知道某列是否可见。但是你仍然可以获得部分信息，因为当调用上面的函数，返回值是 GTK_VISIBILITY_PARTIAL时，它的某一

部分是隐藏的，无法知道到底是被列表的顶部或底部挡住，还是这一行的某列在外边。

你还能改变特定列的前景色和背景色。这可用在当用户选中某行时标记该行。它有两个相关函数：

设置前景颜色：

```
void gtk_clist_set_foreground( GtkCList *clist,
                               gint      row,
                               GdkColor *color );
```

设置背景颜色：

```
void gtk_clist_set_background( GtkCList *clist,
                               gint      row,
                               GdkColor *color );
```

请注意颜色值必须是前面已经分配的。

11.5 向列表中添加行

可以用三种方法添加行。用下面函数可以在前面、后面加入行：

```
gint gtk_clist_prepend( GtkCList *clist,
                        gchar    *text[] );
gint gtk_clist_append( GtkCList *clist,
                       gchar    *text[] );
```

这两个函数返回整数值指明加入到列表中的行的索引号。可以用以下函数在指定位置插入一行：

```
void gtk_clist_insert( GtkCList *clist,
                       gint      row,
                       gchar    *text[] );
```

在这些函数中我们要提供一个指向要插入行的文本数组的指针。指针的个数应该等于列表的列数，如果text[]参数是NULL，这一行的列中就没有文本。当我们想向列表中添加图片时可以这么做。

要注意，行和列的编号都是从0开始的。

用以下函数删除一行：

```
void gtk_clist_remove( GtkCList *clist,
                       gint      row );
```

还有一个函数可以用于删除列表中所有的行。这比对每一行调用一次 gtk_clist_remove函数要来得快。

```
void gtk_clist_clear( GtkCList *clist );
```

还有两个很方便的函数可以用在当列表中要发生很大变化时。因为 GtkCList在发生变化时要重绘自身，所以当列表中内容变化较大时，频繁重绘会让屏幕不停闪烁。最好的办法是先将列表“冻结”，然后更新列表，最后将其“解冻”。

“冻结”构件：

```
void gtk_clist_freeze( GtkCList * clist );
```

将构件“解冻”：

```
void gtk_clist_thaw( GtkCList * clist );
```

11.6 在单元格中设置文本和pixmap图片

单元格可以容纳pixmap图片、文本或同时包含两者。你可以使用以下函数：

```
void gtk_clist_set_text( GtkCList    *clist,
                         gint         row,
                         gint         column,
                         const gchar *text );

void gtk_clist_set_pixmap( GtkCList  *clist,
                           gint       row,
                           gint       column,
                           GdkPixmap *pixmap,
                           GdkBitmap *mask );

void gtk_clist_set_pixtext( GtkCList  *clist,
                            gint       row,
                            gint       column,
                            gchar     *text,
                            guint8     spacing,
                            GdkPixmap *pixmap,
                            GdkBitmap *mask );
```

这些函数应该很容易理解。它们都将要操作的 Clist构件作为第一个参数，第二、三个参数是行或列号，紧接着是要设置的数据。 gtk_clist_set_pixtext 函数中的spacing参数是pixmap图片和文本起始点之间的间距。上面的函数中数据都被复制到 GtkCList构件中。

用以下函数可以读出相应的数据：

```
gint gtk_clist_get_text( GtkCList  *clist,
                         gint       row,
                         gint       column,
                         gchar    **text );
gint gtk_clist_get_pixmap( GtkCList   *clist,
                           gint        row,
                           gint        column,
                           GdkPixmap **pixmap,
                           GdkBitmap **mask );
gint gtk_clist_get_pixtext( GtkCList   *clist,
                            gint        row,
                            gint        column,
                            gchar     **text,
                            guint8     *spacing,
                            GdkPixmap **pixmap,
                            GdkBitmap **mask );
```

返回的指针都是指向存储在构件内部的数据指针，所以不应该被修改或释放。引用的数据没有必要将不感兴趣的数据全部读出。任何返回值指针（除了 GtkCList构件）都可以是NULL。所以如果我们只想从类型为 pixtext的单元格中读出文本，应该像下面这样做，假定clist，row，column都已经存在：

```
gchar *mytext;
gtk_clist_get_pixtext(clist, row, column,
                      &mytext, NULL, NULL, NULL);
```

还有与上面内容相关的几个函数，下面这个函数将返回指定单元格的数据类型：

```
GtkCellType gtk_clist_get_cell_type( GtkCList *clist,
                                     gint      row,
                                     gint      column );
```

返回单元格内的数据类型，返回值可能是以下值之一：

GTK_CELL_EMPTY	单元格内是空的
GTK_CELL_TEXT	单元格内是文本
GTK_CELL_PIXMAP	单元格内是Pixmap图像
GTK_CELL_PIXTEXT	单元格内同时包含文本和图像
GTK_CELL_WIDGET	单元格内包含构件

还有一个函数可以用于设置单元格内水平和垂直方向上的缩排，缩排值是以像素度量的整数值，它可以是正数也可以是负数。

```
void gtk_clist_set_shift( GtkCList *clist,
                          gint      row,
                          gint      column,
                          gint      vertical,
                          gint      horizontal );
```

11.7 存储数据指针

对GtkCList构件来说，可以为一行设置数据指针。指针对用户来说是不可见的。它可方便程序员将某一行与一些其他数据联系起来。

下面函数的意义从字面上就可以理解。

为指定的行设置数据：

```
void gtk_clist_set_row_data( GtkCList *clist,
                             gint      row,
                             gpointer  data );
```

与上面的函数类似，只是当销毁一行时调用由 destory指定的回调函数：

```
void gtk_clist_set_row_data_full( GtkCList         *clist,
                                  gint              row,
                                  gpointer          data,
                                  GtkDestroyNotify  destroy );
```

获取指定行的数据：

```
gpointer gtk_clist_get_row_data( GtkCList *clist,
                                 gint      row );
```

在构件中搜索data所在的行。如果返回-1，则没有找到，或者没有匹配的：

```
gint gtk_clist_find_row_from_data( GtkCList *clist,
                                   gpointer  data );
```

11.8 处理选择

有几个函数可以让我们强行选中或取消一行的选择：

```
void gtk_clist_select_row( GtkCList *clist,
                           gint      row,
                           gint      column );
```

```
void gtk_clist_unselect_row( GtkCList *clist,
                             gint      row,
                             gint      column );
```

还有一个函数使用x、y坐标(例如，从鼠标指针得到坐标)，返回坐标所在的行和列。

```
gint gtk_clist_get_selection_info( GtkCList *clist,
                                   gint      x,
                                   gint      y,
                                   gint     *row,
                                   gint     *column );
```

当我们监测到一些有趣的事件时(也许是鼠标指针的移动，或在列表的某处点击)，可以读出鼠标的坐标值，找出鼠标正在列表的某一格。

11.9 信号

与其他构件一样， GtkCList有一些信号供我们使用。 GtkCList构件是从容器构件GtkContainer派生的，它有容器所具有的一些信号，还有下面这些附加信号：

- select_row：选中一行时引发，该信号传递以下信息，依次是 GtkCList *clist、gint row、gint column、GtkEventButton *event。
- unselect_row：用户对一行取消选择，引发这个信号。传递的信息与上一个信号一样。
- click_column：选中某一列时引发。传递以下信息： GtkCList *clist、gint column。

所以，要将一个回调函数连接到select_row信号上，回调函数应该像下面这样：

```
void select_row_callback(GtkWidget *widget,
                         gint row,
                         gint column,
                         GdkEventButton *event,
                         gpointer data);
```

回调函数用下面的形式连接到信号：

```
gtk_signal_connect(GTK_OBJECT( clist),
                   "select_row"
                   GTK_SIGNAL_FUNC(select_row_callback),
                   NULL);
```

11.10 GtkCList示例

```
/* CLIist示例开始 clist.c */
#include <gtk/gtk.h>
/* 用户点击"Add List按钮时的回调函数*/
void button_add_clicked( gpointer data )
{
    int indx;

    /* 字符串数组，用于加到list中，4行2列*/
    gchar *drink[4][2] = { { "Milk",    "3 Oz" },
                           { "Water",   "6 l" },
                           { "Carrots", "2" },
                           { "Snakes",  "55" } };

    /*下面将文本真正加到list中。对每行添加一次
```

```
     */
    for ( indx=0 ; indx < 4 ; indx++ )
        gtk_clist_append( (GtkCList *) data, drink[indx]);

    return;
}

/*用户点击"Clear List"按钮时的回调函数*/
void button_clear_clicked( gpointer data )
{
    /* 用gtk_clist_clear函数清除列表。比用
     *gtk_clist_remove函数逐行清除要快
     */
    gtk_clist_clear( (GtkCList *) data);

    return;
}

/* 用户点击"Hide/Show title"按钮时的回调函数*/
void button_hide_show_clicked( gpointer data )
{
    /* flag是用于记录可见/不可见状态的标志。0 = 当前不可见 */
    static short int flag = 0;

    if (flag == 0)
    {
        /* 隐藏标题，将flag设置为1 */
        gtk_clist_column_titles_hide((GtkCList *) data);
        flag++;
    }
    else
    {
        /* 形式标题，将flag设置为0 */
        gtk_clist_column_titles_show((GtkCList *) data);
        flag--;
    }

    return;
}

/* 用户选中某一行时的回调函数*/
void selection_made( GtkWidget      *clist,
                     gint            row,
                     gint            column,
                     GdkEventButton *event,
                     gpointer        data )
{
    gchar *text;

    /* 取得存储在被选中的行和列的单元格上的文本
     * 当鼠标点击时，我们用text参数接收一个指针
     */
```

```
    gtk_clist_get_text(GTK_CLIST(clist), row, column, &text);

    /*打印一些关于选中了哪一行的信息*/
    g_print("You selected row %d. More specifically you clicked in "
            "column %d, and the text in this cell is %s\n\n",
            row, column, text);

    return;
}

int main( int    argc,
          gchar *argv[] )
{
    GtkWidget *window;
    GtkWidget *vbox, *hbox;
    GtkWidget *scrolled_window, *clist;
    GtkWidget *button_add, *button_clear, *button_hide_show;
    gchar *titles[2] = { "Ingredients", "Amount" };

    gtk_init(&argc, &argv);

    window=gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_widget_set_usize(GTK_WIDGET(window), 300, 150);

    gtk_window_set_title(GTK_WINDOW(window), "GtkCList Example");
    gtk_signal_connect(GTK_OBJECT(window),
                       "destroy",
                       GTK_SIGNAL_FUNC(gtk_main_quit),
                       NULL);

    vbox=gtk_vbox_new(FALSE, 5);
    gtk_container_set_border_width(GTK_CONTAINER(vbox), 5);
    gtk_container_add(GTK_CONTAINER(window), vbox);
    gtk_widget_show(vbox);

/* 创建一个滚动窗口构件，将GtkCList组装到里面。
 *  这样使得内容超出列表时，可以用滚动条浏览
*/
    scrolled_window = gtk_scrolled_window_new (NULL, NULL);
    gtk_scrolled_window_set_policy (GTK_SCROLLED_WINDOW (scrolled_window),
                                    GTK_POLICY_AUTOMATIC,
    GTK_POLICY_ALWAYS);

    gtk_box_pack_start(GTK_BOX(vbox), scrolled_window, TRUE, TRUE, 0);
    gtk_widget_show (scrolled_window);

    /* 创建GtkCList构件。本例中，我们使用了两列*/
    clist = gtk_clist_new_with_titles( 2, titles);

    /* 当作出选择时，我们要知道选择了哪一个单元格。使用
     * selection_made回调函数，代码在下面可以看见*/
    gtk_signal_connect(GTK_OBJECT(clist), "select_row",
```

```
                    GTK_SIGNAL_FUNC(selection_made),
                    NULL);

/* 不一定要设置边框的阴影，但是效果挺不错 */
gtk_clist_set_shadow_type (GTK_CLIST(clist), GTK_SHADOW_OUT);

/* 很重要的一点，我们设置列宽，让文本能容纳在列中。
 * 注意，列编号是从0开始的
 * 本例中是0和1
 */
gtk_clist_set_column_width (GTK_CLIST(clist), 0, 150);

/* 将GtkCList构件添加到滚动窗口构件中，然后显示 */
gtk_container_add(GTK_CONTAINER(scrolled_window), clist);
gtk_widget_show(clist);

/*创建按钮，把它们加到窗口中
 */
hbox = gtk_hbox_new(FALSE, 0);
gtk_box_pack_start(GTK_BOX(vbox), hbox, FALSE, TRUE, 0);
gtk_widget_show(hbox);

button_add = gtk_button_new_with_label("Add List");
button_clear = gtk_button_new_with_label("Clear List");
button_hide_show = gtk_button_new_with_label("Hide/Show titles");

gtk_box_pack_start(GTK_BOX(hbox), button_add, TRUE, TRUE, 0);
gtk_box_pack_start(GTK_BOX(hbox), button_clear, TRUE, TRUE, 0);
gtk_box_pack_start(GTK_BOX(hbox), button_hide_show, TRUE, TRUE, 0);

/* 为三个按钮的点击设置回调函数 */
gtk_signal_connect_object(GTK_OBJECT(button_add), "clicked",
                          GTK_SIGNAL_FUNC(button_add_clicked),
                          (gpointer) clist);
gtk_signal_connect_object(GTK_OBJECT(button_clear), "clicked",
                          GTK_SIGNAL_FUNC(button_clear_clicked),
                          (gpointer) clist);
gtk_signal_connect_object(GTK_OBJECT(button_hide_show), "clicked",
                          GTK_SIGNAL_FUNC(button_hide_show_clicked),
                          (gpointer) clist);

gtk_widget_show(button_add);
gtk_widget_show(button_clear);
gtk_widget_show(button_hide_show);

/* 界面已经完全设置好了，下面可以显示窗口，进入
 * gtk_mai主循环
 */
gtk_widget_show(window);
gtk_main();
```

```
    return(0);
}

/*示例结束*/
```

编译后，运行效果如图11-1所示。点击Add List按钮，向列表中添加列表项；点击 Clear List按钮，清除所有列表项；点击Hide/Show titles，轮换显示和隐藏列表标题。

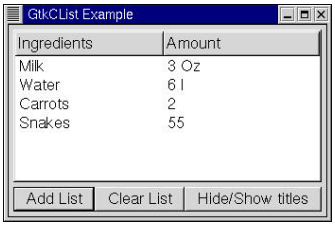

图11-1 GtkCList示例