LinuxFocus article number 303
http://linuxfocus.org

GUI Programming with GTK - part 2

by Özcan Güngör
<ozcangungor(at)netscape.net>

About the author:
I use Linux since 1997. Freedom, flexibility and opensource. These are the properties I like.

Abstract:

In this article, we will discuss boxes and tables. With these, we will be able to locate components in a proper order on windows. In order to understand these articels, you should know the followings about C programming language:

- Variables
- Functions
- Pointers

And additionally, it is recomended to read the previous arctile.

Packing Components with Boxes

Packing componenets means to locate components in proper order on windows. One of the ways to do this in GTK is using boxes. Main idea behind the boxes is to pack the components ordered vertically or horzontally. There are two types of boxes: Horizontal boxes and vertical boxes. Let’s explain these types:

Horizontal Boxes

In this box type, components are packed horizontally. The following function is used to create a horizontal box.

```
gtk_widget *box;
box=gtk_hbox_new(gboolean homogenous, gint spacing);
```

where the parameter *homogeneous* is used to define whether components are distributed homogenously or not: if it is **TRUE**,then components are filling up the whole box such that there is equal distance between them and if it is **FALSE**, then the components are packed side by side. *spacing* is used for setting the mimimun amount of space between components.

Vertical Boxes

In this box type, components are packed vertically. The following function is used to create a vertical box:

```
gtk_widget *box;
box=gtk_vbox_new(gboolean homogenous, gint spacing);
```

Parameters in this function have the meaning as in the horizontal box function.

Common Propertes of Boxes

Adding componenets can be done in two ways: The first one is:

```
gtk_box_pack_start(GtkBox *box,GtkWidget *child,
                   gboolean expand, gboolean fill, guint padding);
```

Using this function we can add components into a box. (on the left side for a horizontal box or at the top of a vertical box) *box* is the box we want to add a component to. *child* is the component to be added. *expand* is used to expand the size of the component such that it uses all space if possible. *padding* is to add additional space on the left and right.

The complementary function to gtk_box_pack_start is gtk_box_pack_end:

```
gtk_box_pack_end(GtkBox *box,GtkWidget *child,
                 gboolean expand, gboolean fill, guint padding);
```

This function lets us add components at the end (right or bottom) of the box. The parameters have the same meaning as in the previous function.

To add a box onto a window, the following function is used:

```
gtk_container_add (GtkContainer *container,GtkWidget *component);
```

container is the window that the box will be added to, *component* is the box to be added. For expample, to add the box we have created above to *window*, following command is used:

```
gtk_container_add(GTK_CONTAINER(window),box);

gtk_box_set_homogeneous (GtkBox *box, gboolean homogenous);
gtk_box_set_spacing(GtkBox *box, gint spacing);
```

The first function above is used to change the property homogenous of a box and the second function is used to change the size of the space of a box. *box* is the box to be changed.

```
gtk_box_set_child_packing(GkBox *box,GtkWidget *shild,
       gboolean expand, gboolean  fill, guint padding,
       GtkPackType packingtype);
```

This function redefines the properties of an already packed component. Parameters have the same meaning as in the function gtk_box_pack_start. *packingtype* can be GTK_PACK_START or GTK_PACK_END. GTK_PACK_START causes the component be packed in the beginning of the box if the component is pack using the function gtk_pack_end. GTK_PACK_END causes the component be packed at the end of the box if the component is packed using the function gtk_pack_start.

To understand better what has explained here, try kutular.c.

Tables

Tables, like in HTML code, help us to fix components in cells. For this, it is enough to create a table which has enough rows and columns. Then we can locate a component in a cell or cell groups (allowed only for cells that are side by side). To create a table, following function is used:

```
GtkWidget *table;
GtkWidget* gtk_table_new(guint  row, guint column, gboolean homogenous);
```

row is the number of rows, *column* is the number of columns.*homogenous* is used to distribute the components homogenously.

The following function is used to add a component to a table:

```
void gtk_table_attach (GtkTable *table, GtkWidget *child,
         guint left_attach, guint left_attach, guint top_attach,
         guint bottom_attach, GtkAttachOptions xoptions,
         GtkAttachOptions yoptions, guint xpadding, guint ypadding);
```

table is the table where components will be added and *child* is the component to be added. *left_attach* is the number of the cell on the left that the component will be put into. *right_attach* is the number of the cell on the right that the component will be put into. *top_attach* is the number of the cell at the top that the component will be located in and *bottom_attach* is the number of the cell at the bottom that the component will be located in. Components may cover in more than one cell.

xoptions and *yoptions* can get three different vaules: GTK_FILL, GTK_EXPAND, GTK_SHRINK. GTK_FILL causes the component to fill the whole cell(s). GTK_EXPAND causes the component be located in the center of the cell(s) and GTK_SHRINK causes the component shrink into the cell(s) dimesions if the component is larger than cell(s).*xoptions* makes these changes only in x axis and

yoptions makes only y axis.

xpadding puts space on the left and rigth of the component along x axis where *ypadding* does it along y axis.

Here is an example code:

```
#include <gtk/gtk.h>

void delete_event( GtkWidget *widget,GdkEvent  *event,gpointer data )
{
    gtk_main_quit ();
}

int main( int   argc,char *argv[] )
{
    GtkWidget *window;
    GtkWidget *button;
    GtkWidget *table;

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

    gtk_signal_connect (GTK_OBJECT (window), "delete_event",
                        GTK_SIGNAL_FUNC (delete_event), NULL);

    table = gtk_table_new (2, 2, TRUE);

    gtk_container_add (GTK_CONTAINER (window), table);

    button = gtk_button_new_with_label ("button 1");
    gtk_table_attach(GTK_TABLE(table), button, 0, 1, 0, 2,GTK_SHRINK,
                     GTK_SHRINK,0,0);
    gtk_widget_show (button);

    button = gtk_button_new_with_label ("button 2");
    gtk_table_attach (GTK_TABLE(table), button, 1, 2, 1, 2,
                      GTK_SHRINK,GTK_SHRINK,0,0);
    gtk_widget_show (button);

    button = gtk_button_new_with_label ("button 3");
    gtk_table_attach (GTK_TABLE(table), button, 1, 2, 0, 1,
                      GTK_SHRINK,GTK_SHRINK,0,0);
    gtk_widget_show (button);

    gtk_widget_show (table);
    gtk_widget_show (window);

    gtk_main ();

    return 0;
}
```

As gtk_table attach has a lot of parameters, a new and short function is created: gtk_table_attach_defaults. This function does the same job but with fewer parameters.

```
void gtk_table_attach_defaults (GtkTable *table,GtkWidget *child,
                guint left_attach, guint right_attach, guint top_attach,
```

```
                guint bottom_attach);
```

Here the parameters have the same meaning.*xoptions* and *yoptions* have the value GTK_FILL|GTK_EXPAND. *xpadding* and *ypadding* have the vaule 0.

The following function is used to change the number of rows and columns of a existing table:

```
void gtk_table_resize(GtkTable *table, guint rows, guint columns);
```

Using the following functions, you can change the spacing value of a row or a column:

```
void gtk_table_set_row_spacing (GtkTable *table, guint row,
                                guint spacing);
void gtk_table_set_col_spacing (GtkTable *table, guint column,
                                guint spacing);
```

The following functions change the spacing values of whole rows or columns:

```
void gtk_table_set_row_spacings (GtkTable *table, guint spacing);
void gtk_table_set_col_spacings (GtkTable *table, guint spacing);
```

The following function changes the homogenous value of a existing table:

```
void gtk_table_set_homogeneous (GtkTable *table, gboolean homogenous);
```

Summary

In this article, we have learnt the means for packing components and then we looked at functions to some properties of boxes and tables. I am always happy to get questions, comments and ideas from readers. Just send me an e-mail....

2005-01-14, generated by lfparser_pdf version 2.51