บทที่ 5

Container Widgets

Container Widget สามารถแบ่งได้หลายชนิดดังนี้

5.1 The EvenBox

การเขียน GTK ใน X Window นั้น บางครั้งไม่สามารถที่จะนำมาปนกันได้จะต้องมีหน้าต่างหลักเพราะจะทำให้ไม่สามารถรับเหตุการและไม่สามารถแก้ไขได้

EventBox เป็นการรวมเอาหลาย ๆ widget ไว้ด้วยกัน โดยที่มี widget หลัก widget เดียวกันและมี widget ย่อย ๆ อยู่ใน widget หลัก โดยมีมีเหตุการใด ซึ่งการใช้ Container Widget แบบนี้จะไม่สามารถดักเหตุการที่เกิดขึ้นได้

การสร้าง Container ชนิด The EvenBox นี้ สามารถทำได้โดยใช้

GtkWidget *gtk_event_box_new(void);

ซึ่งจะสามาถรบรรจุ widget ย่อยได้โดยการ

gtk_container_add (GTK_CONTAINER (event_box), child_widget);

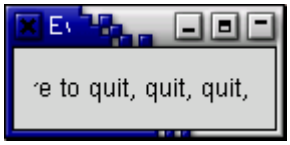
ตัวอย่างโปรแกรม EventBox

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

int main( int argc,
        char *argv[] )
{
   GtkWidget *window;
   GtkWidget *event_box;
   GtkWidget *label;
```

```
gtk_init (&argc, &argv);

window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

gtk_window_set_title (GTK_WINDOW (window), "Event Box");

g_signal_connect (G_OBJECT (window), "destroy",
                  G_CALLBACK (exit), NULL);

gtk_container_set_border_width (GTK_CONTAINER (window), 10);

/* Create an EventBox and add it to our toplevel window */

event_box = gtk_event_box_new ();
gtk_container_add (GTK_CONTAINER (window), event_box);
gtk_widget_show (event_box);

/* Create a long label */

label = gtk_label_new ("Click here to quit, quit, quit, quit, quit");
gtk_container_add (GTK_CONTAINER (event_box), label);
gtk_widget_show (label);

/* Clip it short. */
gtk_widget_set_size_request (label, 110, 20);

/* And bind an action to it */
gtk_widget_set_events (event_box, GDK_BUTTON_PRESS_MASK);
g_signal_connect (G_OBJECT (event_box), "button_press_event",
```

```
                    G_CALLBACK (exit), NULL);

  /* Yet one more thing you need an X window for ... */

  gtk_widget_realize (event_box);
  gdk_window_set_cursor (event_box->window, gdk_cursor_new (GDK_HAND1));

  gtk_widget_show (window);

  gtk_main ();

  return 0;
}
```

ซึ่งจากโปรแกรมดังกล่าวจะได้ผลลัพธ์ดังรูป



5.2 The Alignment widget

The alignment widget จะเป็นการยอมให้วาง widget ใน window โดยที่สามารถกำหนดตำแหน่งและขนาด ได้ด้วยตัวเอง ซึ่งมีรูปแบบการใช้สองรูปแบบ

รูปแบบที่หนึ่ง

```
GtkWidget* gtk_alignment_new(  gfloat xalign,
                               gfloat yalign,
                               gfloat xscale,
                               gfloat yscale );
```

จะเป็นฟังก์ชันการสร้าง widget ชนิด Alignment widget ขึ้นมาใหม่ โดยจะมีรูปแบบการใช้ตามรูปแบบที่หนึ่ง แต่หากมีการเปลี่ยนแปลงสามารถทำได้ตามรูปแบบที่สอง

รูปแบบที่สอง

```
void gtk_alignment_set( GtkAlignment *alignment,
                        gfloat        xalign,
                        gfloat        yalign,
                        gfloat         xscale,
                        gfloat        yscale );
```

ซึ่งค่าใส่ใน xalign, yalign, xscale, yscale จะเป็นค่าที่อยู่ระหว่า 0 – 1 ซึ่งค่า xalign และ yalign จะเป็นค่าที่กำหนดตำแหน่ง ส่วนค่า xscale และ yscale จะเป็นค่าที่กำหนดช่องว่าละหว่างอาร์กิวเมนต์

สามารถได้ด้วยคำสั่ง

```
gtk_container_add (GTK_CONTAINER (alignment), child_widget);
```

5.3 Fixed Container

Fixed Container เป็นการกำหนดที่ที่จะวาง widget โดยมีความสัมพันธ์กับ window ที่จะวาง โดยเมื่อทำการวาง widget ลงใน window แล้ว สามารถเปลี่ยนตำแหน่งได้

การสร้าง Container ชนิด Fixed Container นี้ สามารถทำได้โดยใช้

```
GtkWidget* gtk_fixed_new( void );
```

การกำหนดค่า

```
void gtk_fixed_put(  GtkFixed  *fixed,
                     GtkWidget *widget,
                     gint      x,
                     gint      y );
```

การเปลี่ยนค่า

```
void gtk_fixed_move(    GtkFixed  *fixed,
                        GtkWidget *widget,
                        gint      x,
                        gint      y );
```

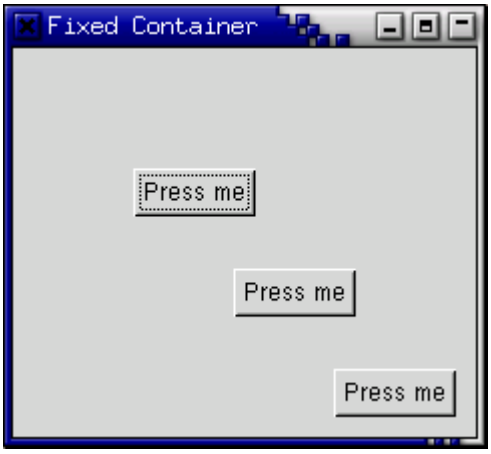
ตัวอย่างโปรแกรม

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

/* I'm going to be lazy and use some global variables to
 * store the position of the widget within the fixed
 * container */
gint x = 50;
gint y = 50;

/* This callback function moves the button to a new position
 * in the Fixed container. */
static void move_button( GtkWidget *widget,
                         GtkWidget *fixed )
{
  x = (x + 30) % 300;
  y = (y + 50) % 300;
  gtk_fixed_move (GTK_FIXED (fixed), widget, x, y);
}

int main( int   argc,
          char *argv[] )
{
  /* GtkWidget is the storage type for widgets */
  GtkWidget *window;
  GtkWidget *fixed;
  GtkWidget *button;
  gint i;

  /* Initialise GTK */
```

```
gtk_init (&argc, &argv);

/* Create a new window */
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW (window), "Fixed Container");

/* Here we connect the "destroy" event to a signal handler */
g_signal_connect (G_OBJECT (window), "destroy",
                  G_CALLBACK (gtk_main_quit), NULL);

/* Sets the border width of the window. */
gtk_container_set_border_width (GTK_CONTAINER (window), 10);

/* Create a Fixed Container */
fixed = gtk_fixed_new ();
gtk_container_add (GTK_CONTAINER (window), fixed);
gtk_widget_show (fixed);

for (i = 1 ; i <= 3 ; i++) {
  /* Creates a new button with the label "Press me" */
  button = gtk_button_new_with_label ("Press me");

  /* When the button receives the "clicked" signal, it will call the
   * function move_button() passing it the Fixed Container as its
   * argument. */
  g_signal_connect (G_OBJECT (button), "clicked",
                    G_CALLBACK (move_button), (gpointer) fixed);

  /* This packs the button into the fixed containers window. */
  gtk_fixed_put (GTK_FIXED (fixed), button, i*50, i*50);
```

```
  /* The final step is to display this newly created widget. */
  gtk_widget_show (button);
 }

 /* Display the window */
 gtk_widget_show (window);

 /* Enter the event loop */
 gtk_main ();

 return 0;
}
```

ซึ่งจากโปรแกรมดังกล่าวจะได้ผลลัพธ์ดังรูป



5.4 Frames

Frames เป็นการจัดเป็นกลุ่ม ให้เห็นว่าเป็นหลุ่มเดียวกัน การใช้ Frames นั้น สามารถที่จะกำหนด labelled ได้

สามารถสร้าง Frames ได้โดยการใช้

```
GtkWidget *gtk_frame_new( const gchar *label );
```

การใส่ labelled ใน Frames

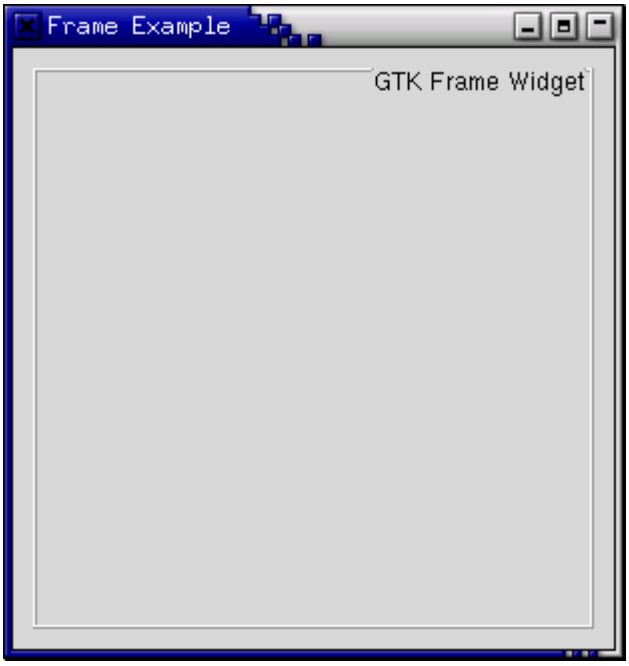
```
void gtk_frame_set_label( GtkFrame    *frame,
                          const gchar *label );
```

ตัวอย่างโปรแกรม

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

int main( int   argc,   char *argv[] )
{
        /* GtkWidget is the storage type for widgets */
        GtkWidget *window;
        GtkWidget *frame;

        /* Initialise GTK */
        gtk_init (&argc, &argv);

        /* Create a new window */
        window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
        gtk_window_set_title (GTK_WINDOW (window), "Frame Example");

        /* Here we connect the "destroy" event to a signal handler */
        g_signal_connect (G_OBJECT (window), "destroy",
                          G_CALLBACK (gtk_main_quit), NULL);

        gtk_widget_set_size_request (window, 300, 300);
        /* Sets the border width of the window. */
        gtk_container_set_border_width (GTK_CONTAINER (window), 10);
```

```
    /* Create a Frame */
    frame = gtk_frame_new (NULL);
    gtk_container_add (GTK_CONTAINER (window), frame);

    /* Set the frame's label */
    gtk_frame_set_label (GTK_FRAME (frame), "GTK Frame Widget");

    /* Align the label at the right of the frame */
    gtk_frame_set_label_align (GTK_FRAME (frame), 1.0, 0.0);

    /* Set the style of the frame */
    gtk_frame_set_shadow_type (GTK_FRAME (frame),
    GTK_SHADOW_ETCHED_OUT);

    gtk_widget_show (frame);

    /* Display the window */
    gtk_widget_show (window);

    /* Enter the event loop */
    gtk_main ();

    return 0;
}
```

ซึ่งจากโปรแกรมดังกล่าวจะได้ผลลัพธ์ดังรูป



5.5 Button Boxes

Button Boxes เป็นการใส่ปุ่มเข้าไปใน widget โดยการใส่จะมีสองประเภทคือแนวตั้งและแนวนอน

สามารถสร้าง Button Boxes ได้โดยการใช้

GtkWidget *gtk_hbutton_box_new(void);

GtkWidget *gtk_vbutton_box_new(void);

การเพิ่มปุ่มเข้าไปใน widget สามารถทำได้โดยการใช้ฟังก์ชัน

gtk_container_add (GTK_CONTAINER (button_box), child_widget);

ตัวอย่างโปรแกรม

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

/* Create a Button Box with the specified parameters */
static GtkWidget *create_bbox(   gint  horizontal,
```

```
                                char *title,
                                gint  spacing,
                                gint  child_w,
                                gint  child_h,
                                gint  layout )
{
        GtkWidget *frame;
        GtkWidget *bbox;
        GtkWidget *button;

        frame = gtk_frame_new (title);

        if (horizontal)
                bbox = gtk_hbutton_box_new ();
        else
                bbox = gtk_vbutton_box_new ();

        gtk_container_set_border_width (GTK_CONTAINER (bbox), 5);
        gtk_container_add (GTK_CONTAINER (frame), bbox);

        /* Set the appearance of the Button Box */
        gtk_button_box_set_layout (GTK_BUTTON_BOX (bbox), layout);
        gtk_box_set_spacing (GTK_BOX (bbox), spacing);
        /*gtk_button_box_set_child_size (GTK_BUTTON_BOX (bbox), child_w, child_h);*/

        button = gtk_button_new_from_stock (GTK_STOCK_OK);
        gtk_container_add (GTK_CONTAINER (bbox), button);

        button = gtk_button_new_from_stock (GTK_STOCK_CANCEL);
        gtk_container_add (GTK_CONTAINER (bbox), button);
```

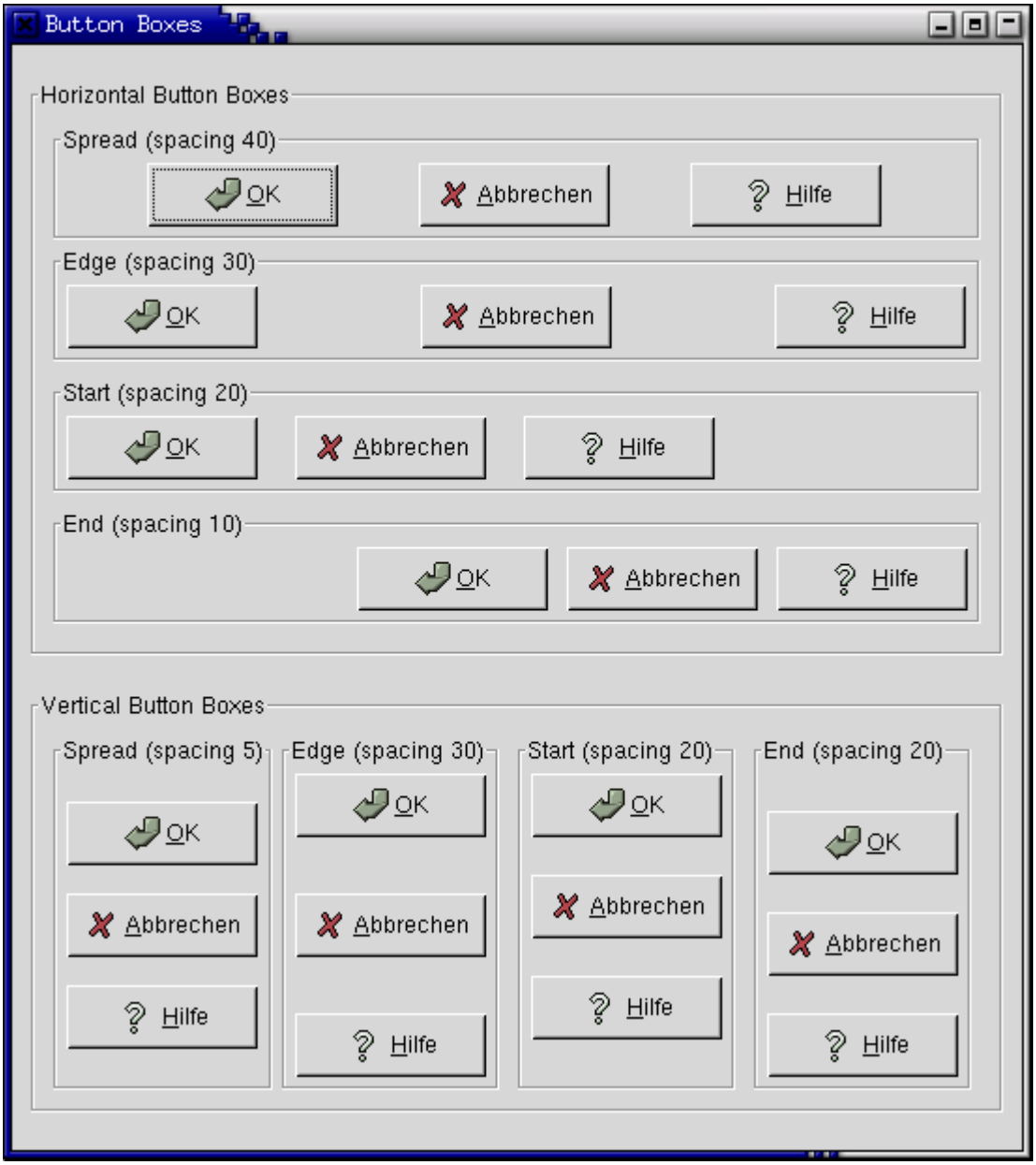
```
        button = gtk_button_new_from_stock (GTK_STOCK_HELP);
        gtk_container_add (GTK_CONTAINER (bbox), button);

        return frame;
}

int main( int   argc,  char *argv[] )
{
        static GtkWidget* window = NULL;
        GtkWidget *main_vbox;
        GtkWidget *vbox;
        GtkWidget *hbox;
        GtkWidget *frame_horz;
        GtkWidget *frame_vert;

        /* Initialize GTK */
        gtk_init (&argc, &argv);

        window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
        gtk_window_set_title (GTK_WINDOW (window), "Button Boxes");

        g_signal_connect (G_OBJECT (window), "destroy",
                        G_CALLBACK (gtk_main_quit),
                        NULL);

        gtk_container_set_border_width (GTK_CONTAINER (window), 10);

        main_vbox = gtk_vbox_new (FALSE, 0);
        gtk_container_add (GTK_CONTAINER (window), main_vbox);
```

```
frame_horz = gtk_frame_new ("Horizontal Button Boxes");
gtk_box_pack_start (GTK_BOX (main_vbox), frame_horz, TRUE, TRUE, 10);

vbox = gtk_vbox_new (FALSE, 0);
gtk_container_set_border_width (GTK_CONTAINER (vbox), 10);
gtk_container_add (GTK_CONTAINER (frame_horz), vbox);

gtk_box_pack_start (GTK_BOX (vbox),
create_bbox (TRUE, "Spread (spacing 40)", 40, 85, 20,
            GTK_BUTTONBOX_SPREAD),
            TRUE, TRUE, 0);

gtk_box_pack_start (GTK_BOX (vbox),
create_bbox (TRUE, "Edge (spacing 30)", 30, 85, 20, GTK_BUTTONBOX_EDGE),
            TRUE, TRUE, 5);

gtk_box_pack_start (GTK_BOX (vbox),
                    create_bbox (TRUE, "Start (spacing 20)",
                    20, 85, 20, GTK_BUTTONBOX_START),
                    TRUE, TRUE, 5);

gtk_box_pack_start (GTK_BOX (vbox),
                    create_bbox (TRUE, "End (spacing 10)",
                    10, 85, 20, GTK_BUTTONBOX_END),
                    TRUE, TRUE, 5);

frame_vert = gtk_frame_new ("Vertical Button Boxes");
gtk_box_pack_start (GTK_BOX (main_vbox), frame_vert, TRUE, TRUE, 10);
```

```
hbox = gtk_hbox_new (FALSE, 0);
gtk_container_set_border_width (GTK_CONTAINER (hbox), 10);
gtk_container_add (GTK_CONTAINER (frame_vert), hbox);

gtk_box_pack_start (GTK_BOX (hbox),
                    create_bbox (FALSE,
                    "Spread (spacing 5)",
                    5, 85, 20, GTK_BUTTONBOX_SPREAD),
                    TRUE, TRUE, 0);

gtk_box_pack_start (GTK_BOX (hbox),
                    create_bbox (FALSE,
                    "Edge (spacing 30)",
                    30, 85, 20, GTK_BUTTONBOX_EDGE),
                    TRUE, TRUE, 5);

gtk_box_pack_start (GTK_BOX (hbox),
                    create_bbox (FALSE,
                    "Start (spacing 20)",
                    20, 85, 20, GTK_BUTTONBOX_START),
                    TRUE, TRUE, 5);

gtk_box_pack_start (GTK_BOX (hbox),
                    create_bbox (FALSE,
                    "End (spacing 20)",
                    20, 85, 20, GTK_BUTTONBOX_END),
                    TRUE, TRUE, 5);

gtk_widget_show_all (window);
```

```
        /* Enter the event loop */
        gtk_main ();

        return 0;
}
```

ซึ่งจากโปรแกรมดังกล่าวจะได้ผลลัพธ์ดังรูป



5.6 Notebooks

Notebooks เป็นการนำ widget มาทำเป็นหน้าสะสมหลาย ๆ หน้า ซ้อนทับกัน ซึ่งการทำเป็น Notebooks นั้น จะสามารถดูข้อมูลเพียงทีละหน้าเดียวเท่านั้น

สามารถสร้าง Notebooks ได้โดย

```
GtkWidget *gtk_notebook_new( void );
```

การเพิ่มหน้า Notebooks ต่อท้ายนั้นสามารถทำได้โดยการใช้ฟังก์ชัน

```
void gtk_notebook_append_page( GtkNotebook *notebook,
                               GtkWidget   *child,
                               GtkWidget   *tab_label );
```

การเพิ่มหน้า Notebooks ก่อนหน้านั้นสามารถทำได้โดยการใช้ฟังก์ชัน

```
void gtk_notebook_prepend_page( GtkNotebook *notebook,
                                GtkWidget   *child,
                                GtkWidget   *tab_label );
```

การลบ Notebooks สามารถทำได้โดยการใช้ฟังก์ชัน

```
void gtk_notebook_remove_page( GtkNotebook *notebook,
                               gint        page_num );
```

ตั้งให้ Noteboos อยู่ที่หน้าใดทำได้โดยใช้ฟังก์ชัน

```
gint gtk_notebook_get_current_page( GtkNotebook *notebook );
```

การเปลี่ยนหน้า Notebooks ก่อนหน้าทำได้โดยการใช้ฟังก์ชั้น

```
void gtk_notebook_prev_page( GtkNoteBook *notebook );
```

การเปลี่ยนหน้า Notebooks ถัดไปทำได้โดยการใช้ฟังก์ชั้น

```
void gtk_notebook_next_page( GtkNoteBook *notebook );
```

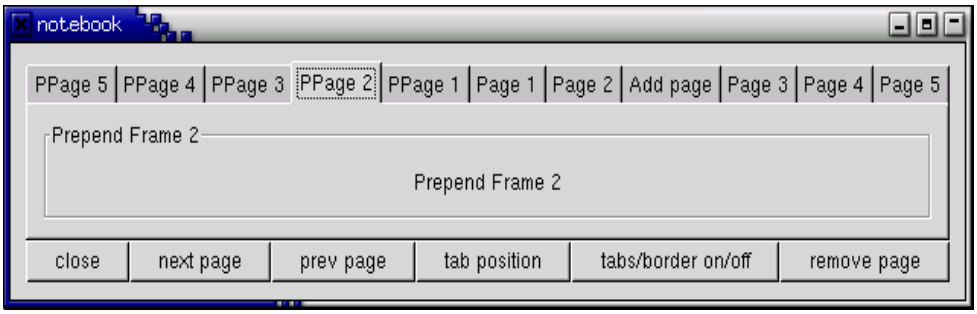
ตัวอย่างโปรแกรม

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

/* This function rotates the position of the tabs */
static void rotate_book( GtkButton   *button, GtkNotebook *notebook )
{
	gtk_notebook_set_tab_pos (notebook, (notebook->tab_pos + 1) % 4);
}

/* Add/Remove the page tabs and the borders */
static void tabsborder_book( GtkButton   *button, GtkNotebook *notebook )
{
	gint tval = FALSE;
	gint bval = FALSE;
	if (notebook->show_tabs == 0)
		tval = TRUE;
	if (notebook->show_border == 0)
		bval = TRUE;

	gtk_notebook_set_show_tabs (notebook, tval);
	gtk_notebook_set_show_border (notebook, bval);
}

/* Remove a page from the notebook */
static void remove_book( GtkButton   *button, GtkNotebook *notebook )
{
	gint page;
```

```
        page = gtk_notebook_get_current_page (notebook);
        gtk_notebook_remove_page (notebook, page);
        /* Need to refresh the widget --
         *  This forces the widget to redraw itself.
         */
        gtk_widget_queue_draw (GTK_WIDGET (notebook));
}

static gboolean delete( GtkWidget *widget,  GtkWidget *event,  gpointer   data )
{
        gtk_main_quit ();
        return  FALSE;
}

int main( int argc,  char *argv[] )
{
        GtkWidget *window;
        GtkWidget *button;
        GtkWidget *table;
        GtkWidget *notebook;
        GtkWidget *frame;
        GtkWidget *label;
        GtkWidget *checkbutton;
        int i;
        char bufferf[32];
        char bufferl[32];

        gtk_init (&argc, &argv);

        window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
```

```
g_signal_connect (G_OBJECT (window), "delete_event",
                    G_CALLBACK (delete), NULL);

gtk_container_set_border_width (GTK_CONTAINER (window), 10);

table = gtk_table_new (3, 6, FALSE);
gtk_container_add (GTK_CONTAINER (window), table);

/* Create a new notebook, place the position of the tabs */
notebook = gtk_notebook_new ();
gtk_notebook_set_tab_pos (GTK_NOTEBOOK (notebook), GTK_POS_TOP);
gtk_table_attach_defaults (GTK_TABLE (table), notebook, 0, 6, 0, 1);
gtk_widget_show (notebook);

/* Let's append a bunch of pages to the notebook */
for (i = 0; i < 5; i++)
{
        sprintf(bufferf, "Append Frame %d", i + 1);
        sprintf(bufferl, "Page %d", i + 1);

        frame = gtk_frame_new (bufferf);
        gtk_container_set_border_width (GTK_CONTAINER (frame), 10);
        gtk_widget_set_size_request (frame, 100, 75);
        gtk_widget_show (frame);

        label = gtk_label_new (bufferf);
        gtk_container_add (GTK_CONTAINER (frame), label);
        gtk_widget_show (label);

        label = gtk_label_new (bufferl);
```

```
        gtk_notebook_append_page (GTK_NOTEBOOK (notebook), frame, label);
}

/* Now let's add a page to a specific spot */
checkbutton = gtk_check_button_new_with_label ("Check me please!");
gtk_widget_set_size_request (checkbutton, 100, 75);
gtk_widget_show (checkbutton);

label = gtk_label_new ("Add page");
gtk_notebook_insert_page (GTK_NOTEBOOK (notebook), checkbutton, label, 2);

/* Now finally let's prepend pages to the notebook */
for (i = 0; i < 5; i++) {
        sprintf (bufferf, "Prepend Frame %d", i + 1);
        sprintf (bufferl, "PPage %d", i + 1);

        frame = gtk_frame_new (bufferf);
        gtk_container_set_border_width (GTK_CONTAINER (frame), 10);
        gtk_widget_set_size_request (frame, 100, 75);
        gtk_widget_show (frame);

        label = gtk_label_new (bufferf);
        gtk_container_add (GTK_CONTAINER (frame), label);
        gtk_widget_show (label);

        label = gtk_label_new (bufferl);
        gtk_notebook_prepend_page (GTK_NOTEBOOK (notebook), frame, label);
}

/* Set what page to start at (page 4) */
```

```
gtk_notebook_set_current_page (GTK_NOTEBOOK (notebook), 3);

/* Create a bunch of buttons */
button = gtk_button_new_with_label ("close");
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                          G_CALLBACK (delete), NULL);
gtk_table_attach_defaults (GTK_TABLE (table), button, 0, 1, 1, 2);
gtk_widget_show (button);

button = gtk_button_new_with_label ("next page");
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                          G_CALLBACK (gtk_notebook_next_page),
                          G_OBJECT (notebook));
gtk_table_attach_defaults (GTK_TABLE (table), button, 1, 2, 1, 2);
gtk_widget_show (button);

button = gtk_button_new_with_label ("prev page");
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                          G_CALLBACK (gtk_notebook_prev_page),
                          G_OBJECT (notebook));
gtk_table_attach_defaults (GTK_TABLE (table), button, 2, 3, 1, 2);
gtk_widget_show (button);

button = gtk_button_new_with_label ("tab position");
g_signal_connect (G_OBJECT (button), "clicked",
                  G_CALLBACK (rotate_book),
                  (gpointer) notebook);
gtk_table_attach_defaults (GTK_TABLE (table), button, 3, 4, 1, 2);
gtk_widget_show (button);
```

```
        button = gtk_button_new_with_label ("tabs/border on/off");
        g_signal_connect (G_OBJECT (button), "clicked",
                          G_CALLBACK (tabsborder_book),
                          (gpointer) notebook);
        gtk_table_attach_defaults (GTK_TABLE (table), button, 4, 5, 1, 2);
        gtk_widget_show (button);

        button = gtk_button_new_with_label ("remove page");
        g_signal_connect (G_OBJECT (button), "clicked",
        G_CALLBACK (remove_book),
                         (gpointer) notebook);
        gtk_table_attach_defaults (GTK_TABLE (table), button, 5, 6, 1, 2);
        gtk_widget_show (button);

        gtk_widget_show (table);
        gtk_widget_show (window);

        gtk_main ();

        return 0;
}
```