บทที่ 1

การสร้าง Packing Widgets

เมื่อคุณต้องการสร้างโปรแกรมประยุกต์ด้วย GTK คุณจะต้องนำวัตถุทุกสิ่งที่สร้างขึ้นมานั้นใส่ลงในหน้าต่างเพียงอันเดียว จากตัวอย่างโปรแกรมแรก เราใช้คำสั่ง gtk_container_add() ในการบรรจุสิ่งของต่างๆ ที่สร้างขึ้นลงในหน้าต่าง คำสั่งดังกล่าวเรียกว่า “การบรรจุ (pack)” นั่นเอง เมื่อคุณต้องการใส่วัตถุทั้งหลายที่สร้างขึ้นลงในหน้าต่าง วิธีที่จะควบคุมตำแหน่งในการจัดวางวัตถุต่างๆ นั้น จะได้เรียนรู้กันในบทนี้

1.1 Theory of Packing Boxes

โดยส่วนใหญ่แล้วการบรรจุวัตถุต่างๆ นั้นมักจะทำโดยการสร้าง box เราสามารถทำการบรรจุ Widget เหล่านี้ได้สองแบบด้วยกันคือ horizontal box และ vertical box การบรรจุวัตถุแบบ horizontal นั้นจะใส่วัตถุจากด้านซ้ายไปด้านขวา หรือ ด้านขวามาด้านซ้าย ขึ้นอยู่กับการเรียกใช้ ส่วนการบรรจุวัตถุแบบ vertical วัตถุจะถูกบรรจุจากด้านบนลงล่างเสมอ คุณอาจจะใช้การบรรจุทั้งสองแบบร่วมกันได้

ในการสร้าง horizontal box นั้นจะใช้คำสั่ง gtk_hbox_new() ส่วนในการสร้าง vertical box ก็จะใช้คำสั่ง gtk_vbox_new() ฟังก์ชัน gtk_box_pack_start() กับ gtk_box_pack_end() ใช้สำหรับบรรจุวัตถุต่างๆ ที่สร้างขึ้น โดยฟังก์ชัน gtk_box_pack_start() จะทำการบรรจุจากด้านบนสุดลงสู่ด้านล่างถ้าเป็นการบรรจุวัตถุลงใน vbox แต่จะบรรจุวัตถุจากด้านซ้ายไปขวาถ้าบรรจุวัตถุลงใน hbox ส่วนฟังก์ชัน gtk_box_pack_end() จะทำตรงกันข้ามกับฟังก์ชัน gtk_box_pack_start() กล่าวคือ ฟังก์ชัน gtk_box_pack_end() จะทำการบรรจุวัตถุต่างๆ ที่สร้างขึ้นจากด้านบนลงด้านล่างถ้าเป็นการบรรจุวัตถุลงใน vbox และจะบรรจุวัตถุจากด้านขวามาด้านซ้ายถ้าเป็นการบรรจุแบบ hbox การใช้ฟังก์ชันนี้ จะอนุญาตให้จัดขอบชิดด้านขวาหรือซ้ายของวัตถุนั้น ส่วนใหญ่จะใช้ฟังก์ชัน gtk_box_pack_start() ในโปรแกรมตัวอย่างต่างๆ ที่ทำขึ้น วัตถุต่างๆ อาจจะถูกบรรจุใน Container อื่นๆ หรือ Widget อื่นๆ ก็ได้ อันที่จริงแล้ววัตถุส่วนใหญ่ จะถูกบรรจุอยู่ในตัวมันเองอยู่แล้ว เช่น ในการสร้างปุ่มนั้น ข้อความที่บรรจุอยู่ภายในปุ่มก็เป็นการบรรจุของตัวมันเอง แต่เราจะเรียกข้อความภายในปุ่มนั้นว่า label ที่อยู่ภายในปุ่ม

1.2 Details of Boxes

สิ่งที่จะกล่าวถึงต่อไปในหัวข้อนี้คือ option ต่างๆ ที่จำเป็นต้องใช้ในการบรรจุวัตถุ เพื่อให้การบรรจุวัตถุเหล่านั้นเกิดความยืดหยุ่นมากขึ้นดังนี้

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

พารามิเตอร์ตัวแรกเป็น box ที่ต้องการจะนำวัตถุไปใส่ พารามิเตอร์ตัวที่สองเป็นชื่อของวัตถุที่ต้องการนำมาใส่ลงใน box พารามิเตอร์ตัวที่สามเป็นการกำหนดว่าต้องการขยายขนาดระยะห่างของวัตถุอัตโนมัติหรือไม่ โดยพารามิเตอร์ตัวนี้จะมีค่าเป็น TRUE หรือ FALSE เท่านั้น พารามิเตอร์ตัวที่สี่เป็นการกำหนดว่าจะให้มีการขยายขนาดของวัตถุแบบอัตโนมัติหรือไม่ โดยพารามิเตอร์ตัวนี้จะมีค่าเป็น TRUE หรือ FALSE เช่นกัน ส่วนพารามิเตอร์ตัวสุดท้ายเป็นการระบุค่าขนาดของกรอบของวัตถุ ว่าต้องการให้วัตถุนั้นมีขนาดของกรอบมากน้อยเพียงใด

เมื่อจะทำการสร้าง box ใหม่ขึ้นมาทำได้โดยใช้ฟังก์ชันดังนี้

```
GtkWidget *gtk_hbox_new(gboolean homogeneous, gint spacing);
GtkWidget *gtk_vbox_new(gboolean homogeneous, gint spacing);
```

พารามิเตอร์ตัวแรกของฟังก์ชันทั้งสองเป็นการควบคุมขนาดของวัตถุที่จะบรรจุภายใน box โดยที่พารามิเตอร์ตัวนี้จะมีค่าเป็น TRUE หรือ FALSE ส่วนพารามิเตอร์ตัวที่สองเป็นตัวระบุช่องว่างระหว่างวัตถุ

ตัวอย่างโปรแกรมการ Pack

ในหัวข้อนี้จะแสดงตัวอย่างโปรแกรมการบรรจุวัตถุลงสู่ box ดังต่อไปนี้

```
/* example-start packbox packbox.c */
#include <stdio.h>
```

```
#include <stdlib.h>
#include "gtk/gtk.h"
static gboolean delete_event( GtkWidget *widget,
                              GdkEvent  *event,
                              gpointer   data )
{
   gtk_main_quit ();
   return FALSE;
}

/* Make a new hbox filled with button-labels. Arguments for the
 * variables we're interested are passed in to this function.
 * We do not show the box, but do show everything inside. */
static GtkWidget *make_box( gboolean homogeneous,
                            gint     spacing,
                            gboolean expand,
                            gboolean fill,
                            guint    padding )
{
   GtkWidget *box;
   GtkWidget *button;
   char padstr[80];

   /* Create a new hbox with the appropriate homogeneous
    * and spacing settings */
   box = gtk_hbox_new (homogeneous, spacing);

   /* Create a series of buttons with the appropriate settings */
   button = gtk_button_new_with_label ("gtk_box_pack");
   gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
```

```
gtk_widget_show (button);

button = gtk_button_new_with_label ("(box,");
gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
gtk_widget_show (button);

button = gtk_button_new_with_label ("button,");
gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
gtk_widget_show (button);

/* Create a button with the label depending on the value of   * expand. */
if (expand == TRUE)
        button = gtk_button_new_with_label ("TRUE,");
else
        button = gtk_button_new_with_label ("FALSE,");

gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
gtk_widget_show (button);



/* This is the same as the button creation for "expand"
 * above, but uses the shorthand form. */
button = gtk_button_new_with_label (fill ? "TRUE," : "FALSE,");
gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
gtk_widget_show (button);

sprintf (padstr, "%d);", padding);

button = gtk_button_new_with_label (padstr);
```

```
   gtk_box_pack_start (GTK_BOX (box), button, expand, fill, padding);
   gtk_widget_show (button);

   return box;
}
int main( int   argc,
        char *argv[])
{
   GtkWidget *window;
   GtkWidget *button;
   GtkWidget *box1;
   GtkWidget *box2;
   GtkWidget *separator;
   GtkWidget *label;
   GtkWidget *quitbox;
   int which;



   /* Our init, don't forget this! :) */
   gtk_init (&argc, &argv);

   if (argc != 2) {
            fprintf (stderr, "usage: packbox num, where num is 1, 2, or 3.\n");
            /* This just does cleanup in GTK and exits with an exit status of 1. */
            exit (1);
   }

   which = atoi (argv[1]);
```

```
/* Create our window */
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

/* You should always remember to connect the delete_event signal
 * to the main window. This is very important for proper intuitive
 * behavior */
g_signal_connect (G_OBJECT (window), "delete_event",
                  G_CALLBACK (delete_event), NULL);
gtk_container_set_border_width (GTK_CONTAINER (window), 10);

/* We create a vertical box (vbox) to pack the horizontal boxes into.
 * This allows us to stack the horizontal boxes filled with buttons one
 * on top of the other in this vbox. */
box1 = gtk_vbox_new (FALSE, 0);



/* which example to show. These correspond to the pictures above. */
switch (which) {
case 1:
        /* create a new label. */
        label = gtk_label_new ("gtk_hbox_new (FALSE, 0);");

        /* Align the label to the left side.  We'll discuss this function and
         * others in the section on Widget Attributes. */
        gtk_misc_set_alignment (GTK_MISC (label), 0, 0);

        /* Pack the label into the vertical box (vbox box1).  Remember that
         * widgets added to a vbox will be packed one on top of the other in
         * order. */
```

```
    gtk_box_pack_start (GTK_BOX (box1), label, FALSE, FALSE, 0);

    /* Show the label */
    gtk_widget_show (label);

    /* Call our make box function - homogeneous = FALSE, spacing = 0,
     * expand = FALSE, fill = FALSE, padding = 0 */
    box2 = make_box (FALSE, 0, FALSE, FALSE, 0);
    gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
    gtk_widget_show (box2);

    /* Call our make box function - homogeneous = FALSE, spacing = 0,
     * expand = TRUE, fill = FALSE, padding = 0 */
    box2 = make_box (FALSE, 0, TRUE, FALSE, 0);
    gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
    gtk_widget_show (box2);

    /* Args are: homogeneous, spacing, expand, fill, padding */
    box2 = make_box (FALSE, 0, TRUE, TRUE, 0);
    gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
    gtk_widget_show (box2);

    /* Creates a separator, we'll learn more about these later,
     * but they are quite simple. */
    separator = gtk_hseparator_new ();

/* Pack the separator into the vbox. Remember each of these
 * widgets is being packed into a vbox, so they'll be stacked
     * vertically. */
    gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 5);
```

```
        gtk_widget_show (separator);

        /* Create another new label, and show it. */
        label = gtk_label_new ("gtk_hbox_new (TRUE, 0);");
        gtk_misc_set_alignment (GTK_MISC (label), 0, 0);
        gtk_box_pack_start (GTK_BOX (box1), label, FALSE, FALSE, 0);
        gtk_widget_show (label);

        /* Args are: homogeneous, spacing, expand, fill, padding */
        box2 = make_box (TRUE, 0, TRUE, FALSE, 0);
        gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
        gtk_widget_show (box2);

        /* Args are: homogeneous, spacing, expand, fill, padding */
        box2 = make_box (TRUE, 0, TRUE, TRUE, 0);
        gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
        gtk_widget_show (box2);

        /* Another new separator. */
        separator = gtk_hseparator_new ();
        /* The last 3 arguments to gtk_box_pack_start are:
         * expand, fill, padding. */
        gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 5);
        gtk_widget_show (separator);
        break;

    case 2:

        /* Create a new label, remember box1 is a vbox as created
         * near the beginning of main() */
```

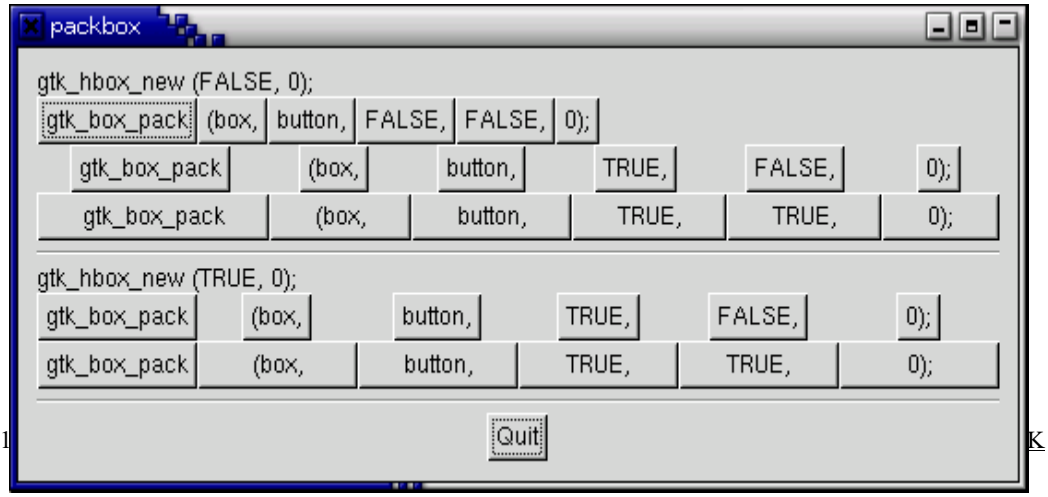
```
label = gtk_label_new ("gtk_hbox_new (FALSE, 10);");
gtk_misc_set_alignment (GTK_MISC (label), 0, 0);
gtk_box_pack_start (GTK_BOX (box1), label, FALSE, FALSE, 0);
gtk_widget_show (label);

/* Args are: homogeneous, spacing, expand, fill, padding */
box2 = make_box (FALSE, 10, TRUE, FALSE, 0);
gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
gtk_widget_show (box2);

/* Args are: homogeneous, spacing, expand, fill, padding */
box2 = make_box (FALSE, 10, TRUE, TRUE, 0);
gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
gtk_widget_show (box2);

separator = gtk_hseparator_new ();

/* The last 3 arguments to gtk_box_pack_start are:
 * expand, fill, padding. */
gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 5);
gtk_widget_show (separator);

label = gtk_label_new ("gtk_hbox_new (FALSE, 0);");
gtk_misc_set_alignment (GTK_MISC (label), 0, 0);
gtk_box_pack_start (GTK_BOX (box1), label, FALSE, FALSE, 0);
gtk_widget_show (label);
/* Args are: homogeneous, spacing, expand, fill, padding */
box2 = make_box (FALSE, 0, TRUE, FALSE, 10);
gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
gtk_widget_show (box2);
```

```
        /* Args are: homogeneous, spacing, expand, fill, padding */
        box2 = make_box (FALSE, 0, TRUE, TRUE, 10);
        gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
        gtk_widget_show (box2);

        separator = gtk_hseparator_new ();
        /* The last 3 arguments to gtk_box_pack_start are: expand, fill, padding. */
        gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 5);
        gtk_widget_show (separator);
        break;

case 3:

  /* This demonstrates the ability to use gtk_box_pack_end() to
        * right justify widgets. First, we create a new box as before. */
        box2 = make_box (FALSE, 0, FALSE, FALSE, 0);
        /* Create the label that will be put at the end. */
        label = gtk_label_new ("end");
        /* Pack it using gtk_box_pack_end(), so it is put on the right
        * side of the hbox created in the make_box() call. */
        gtk_box_pack_end (GTK_BOX (box2), label, FALSE, FALSE, 0);
        /* Show the label. */
        gtk_widget_show (label);

        /* Pack box2 into box1 (the vbox remember ? :) */
        gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, FALSE, 0);
        gtk_widget_show (box2);

        /* A separator for the bottom. */
```

```
        separator = gtk_hseparator_new ();
        /* This explicitly sets the separator to 400 pixels wide by 5 pixels
         * high. This is so the hbox we created will also be 400 pixels wide,
         * and the "end" label will be separated from the other labels in the
         * hbox. Otherwise, all the widgets in the hbox would be packed as
         * close together as possible. */
        gtk_widget_set_size_request (separator, 400, 5);
        /* pack the separator into the vbox (box1) created near the start
         * of main() */
        gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 5);
        gtk_widget_show (separator);
}

/* Create another new hbox.. remember we can use as many as we need! */
quitbox = gtk_hbox_new (FALSE, 0);



/* Our quit button. */
button = gtk_button_new_with_label ("Quit");

/* Setup the signal to terminate the program when the button is clicked */
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                                G_CALLBACK (gtk_main_quit),
                                G_OBJECT (window));
/* Pack the button into the quitbox.
 * The last 3 arguments to gtk_box_pack_start are:
 * expand, fill, padding. */
gtk_box_pack_start (GTK_BOX (quitbox), button, TRUE, FALSE, 0);
/* pack the quitbox into the vbox (box1) */
gtk_box_pack_start (GTK_BOX (box1), quitbox, FALSE, FALSE, 0);
```

```
    /* Pack the vbox (box1) which now contains all our widgets, into the
     * main window. */
    gtk_container_add (GTK_CONTAINER (window), box1);

    /* And show everything left */
    gtk_widget_show (button);
    gtk_widget_show (quitbox);

    gtk_widget_show (box1);
    /* Showing the window last so everything pops up at once. */
    gtk_widget_show (window);

    /* And of course, our main function. */
    gtk_main ();



    /* Control returns here when gtk_main_quit() is called, but not when
     * exit() is used. */

    return 0;
}
```

เมื่อทำการคอมไพล์เรียบร้อยแล้ว ทำการรันโปรแกรมโดยพิมพ์คำสั่ง ./a.out ตามด้วยตัวเลขเช่น ./a.out 2 จะได้ผลดังนี้

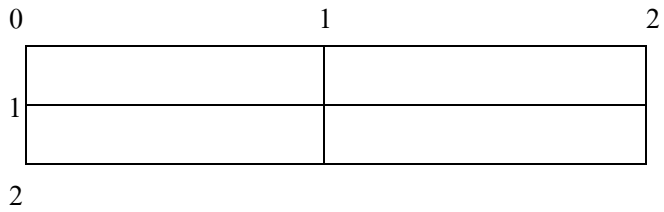


1.3 Packing Using Tables

จากการที่ได้ทำการศึกษาการบรรจุวัตถุต่างๆ ลงสู่ box กันแล้ว ในหัวข้อนี้จะเป็นการศึกษาการบรรจุวัตถุอีกแบบหนึ่ง คือการบรรจุวัตถุโดยใช้ตาราง ซึ่งการใช้ตารางในการบรรจุวัตถุนั้น จะทำให้ทราบตำแหน่งที่แน่นอนของวัตถุที่จะทำการบรรจุลงไป ฟังก์ชันแรกที่จะแนะนำในการสร้างตารางคือ

```
GtkWidget *gtk_table_new(guint rows, guint columns, gboolean homogeneous);
```

พารามิเตอร์ตัวแรกเป็นจำนวนแถวของตาราง พารามิเตอร์ตัวที่สองเป็นจำนวนคอลัมน์ พารามิเตอร์ตัวสุดท้ายเป็นการกำหนดขนาดให้กับบตาราง ถ้ามีค่าเป็น TRUE จะสามารถทำการปรับขนาดของตารางตามขนาดของวัตถุที่ใหญ่ที่สุดที่บรรจุภายในตาราง ถ้ามีค่าเป็น FALSE จะเป็นการบังคับโดยความสูงที่สุดขอแถวใกล้เคียงและความกว้างสุดของวัตถุที่อยู่ภายในคอลัมน์ใกล้เคียง ตัวเลขที่ใช้กำหนดขนาดของแถวและคอลัมน์จะเริ่มต้นที่ 0 ถ้าต้องการสร้างตารางที่มี 2 แถว 2 คอลัมน์ จะได้รูปตารางดังต่อไปนี้



ในการบรรจุวัตถุต่างๆ ที่สร้างขึ้นลงในตารางสามารถทำได้โดยใช้ฟังก์ชันดังนี้

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

พารามิเตอร์ตัวแรกเป็นชื่อตารางที่ทำการสร้างขึ้นมา พารามิเตอร์ตัวที่สองเป็นชื่อของวัตถุที่ต้องการนำมาบรรจุลงในตาราง ส่วนพารามิเตอร์ตัวที่ 3 – 6 เป็นตำแหน่งที่ต้องการวางวัตถุ พารามิเตอร์ตัวที่ 7 และ 8 มีไว้สำหรับระบุ option เพิ่มเติมโดย option ต่างๆ ที่จะได้อธิบายดังนี้

GTK_FILL เป็น option ที่ใช้ปรับขนาดของวัตถุให้เหมาะกับตาราง

GTK_SHRINK เป็น option ที่ถ้าตารางมีขนาดเล็กเกินกว่าที่จะบรรจุวัตถุที่สร้างขึ้นได้ option นี้จะทำการขยายหน้าต่างหลักให้ตารางนั้นใหญ่พอที่จะแสดงวัตถุนั้นได้

GTK_EXPAND เป็น option ที่กำหนดให้ตารางขยายขนาดให้เหมาะกับหน้าต่างหลัก

พารามิเตอร์สองตัวสุดท้ายเป็นระยะห่างของตาราง ฟังก์ชันที่ใช้ในการบรรจุวัตถุลงตารางยังมีอีกฟังก์ชันหนึ่งซึ่งเป็นฟังก์ชันที่ย่อมาจากฟังก์ชัน gtk_table_attach()

```
void gtk_table_attach_defaults(GtkTable *table, GtkWidget *widget,
                               guint left_attach, guint right_attach,
                               guint top_attach, guint bottom_attach);
```

ฟังก์ชันนี้จะเหมือนกับฟังก์ชัน gtk_table_attach() แต่จะละในส่วนของ option ไว้

นอกจากนี้ยังสามารถใช้ฟังก์ชัน gtk_table_set_row_spacing() และ gtk_table_set_col_spacing() ในการกำหนดระยะห่างระหว่างตารางแต่ละช่องได้อีกด้วย โดยที่ฟังก์ชันทั้งสองมีรายละเอียดต่างๆ ดังนี้

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

ฟังก์ชันทั้งสองใช้สำหรับกำหนดระยะห่างของตารางโดยระบุแถวหรือคอลัมน์ที่ต้องการจะกำหนดระยะห่าง

```
void gtk_table_set_row_spacing(GtkTable *table, gint spacing);
void gtk_table_set_col_spacing(GtkTable *table, gint spacing);
```

ฟังก์ชันทั้งสองใช้สำหรับกำหนดระยะห่างของตารางทั้งตาราง

ตัวอย่างโปรแกรมการ Pack โดยใช้ตาราง

ในหัวข้อนี้จะแสดงตัวอย่างโปรแกรมการบรรจุวัตถุลงสู่ตารางดังต่อไปนี้

```
#include <gtk/gtk.h>
/* Our callback.
 * The data passed to this function is printed to stdout */
static void callback( GtkWidget *widget,
                      gpointer   data )
{
   g_print ("Hello again - %s was pressed\n", (char *) data);
}
/* This callback quits the program */
static gboolean delete_event( GtkWidget *widget,
                              GdkEvent  *event,
                              gpointer   data )
{
   gtk_main_quit ();
   return FALSE;
}

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

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

```
/* Create a new window */
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

/* Set the window title */
gtk_window_set_title (GTK_WINDOW (window), "Table");

/* Set a handler for delete_event that immediately
 * exits GTK. */
g_signal_connect (G_OBJECT (window), "delete_event",
                G_CALLBACK (delete_event), NULL);

/* Sets the border width of the window. */
gtk_container_set_border_width (GTK_CONTAINER (window), 20);

/* Create a 2x2 table */
table = gtk_table_new (2, 2, TRUE);

/* Put the table in the main window */
gtk_container_add (GTK_CONTAINER (window), table);

/* Create first button */
button = gtk_button_new_with_label ("button 1");

/* When the button is clicked, we call the "callback" function
 * with a pointer to "button 1" as its argument */
g_signal_connect (G_OBJECT (button), "clicked",
                G_CALLBACK (callback), (gpointer) "button 1");

/* Insert button 1 into the upper left quadrant of the table */
```

```
gtk_table_attach_defaults (GTK_TABLE (table), button, 0, 1, 0, 1);

gtk_widget_show (button);

/* Create second button */

button = gtk_button_new_with_label ("button 2");

/* When the button is clicked, we call the "callback" function
 * with a pointer to "button 2" as its argument */
g_signal_connect (G_OBJECT (button), "clicked",
            G_CALLBACK (callback), (gpointer) "button 2");
/* Insert button 2 into the upper right quadrant of the table */
gtk_table_attach_defaults (GTK_TABLE (table), button, 1, 2, 0, 1);

gtk_widget_show (button);

/* Create "Quit" button */
button = gtk_button_new_with_label ("Quit");

/* When the button is clicked, we call the "delete_event" function
 * and the program exits */
g_signal_connect (G_OBJECT (button), "clicked",
            G_CALLBACK (delete_event), NULL);

/* Insert the quit button into the both
 * lower quadrants of the table */
gtk_table_attach_defaults (GTK_TABLE (table), button, 0, 2, 1, 2);

gtk_widget_show (button);
```

```
    gtk_widget_show (table);
    gtk_widget_show (window);

    gtk_main ();

    return 0;
}
```

เมื่อทำการคอมไพล์และรันโปรแกรมโดยพิมพ์คำสั่ง ./a.out จะได้ผลลัพธ์ดังต่อไปนี้