บทที่ 3

การสร้าง Button Widget

3.1 Normal Buttons

มีหลายวิธีด้วยกันในการสร้างปุ่มแบบทั่วไป ถ้าต้องการสร้างปุ่มที่มี label ใช้ฟังก์ชัน gtk_button_new_with_label() หรือ gtk_button_new_with_mnemonic() และใช้ฟังก์ชัน gtk_button_new_from_stock() เพื่อสร้างปุ่มที่มีภาพและข้อความจาก stock หรืออาจจะเป็น gtk_button_new() เพื่อที่จะสร้างปุ่มที่ไม่มีข้อความใด ๆ วิธีการที่จะใส่ข้อความหรือรูปภาพในปุ่มทำได้โดยการสร้าง box ขึ้นมาใหม่แล้วใส่ object เข้าไปใน box โดยใช้ฟังก์ชัน gtk_box_pack_start() จากนั้นใช้ฟังก์ชัน gtk_container_add() เพื่อที่จะ pack box เข้าไปไว้ในปุ่ม

ตัวอย่างของการใช้ฟังก์ชัน gtk_button_new() แล้วใส่ภาพและข้อความเข้าไปในปุ่ม ทำได้ดังนี้

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

/* Create a new hbox with an image and a label packed into it
 * and return the box. */

static GtkWidget *xpm_label_box( gchar     *xpm_filename,
                                 gchar     *label_text )
{
   GtkWidget *box;
   GtkWidget *label;
   GtkWidget *image;

   /* สร้าง box เพื่อจะใช้บรรจุข้อความและรูปภาพ */
   box = gtk_hbox_new (FALSE, 0);
```

```
   gtk_container_set_border_width (GTK_CONTAINER (box), 2);

   /* เรียกใช้รูปภาพที่มีอยู่ในไฟล์ */
   image = gtk_image_new_from_file (xpm_filename);

   /* สร้าง label พร้อมด้วยข้อความ */
   label = gtk_label_new (label_text);

   /* Pack รูปภาพและ label เข้าไปใน box */
   gtk_box_pack_start (GTK_BOX (box), image, FALSE, FALSE, 3);
   gtk_box_pack_start (GTK_BOX (box), label, FALSE, FALSE, 3);

   gtk_widget_show (image);
   gtk_widget_show (label);

   return box;
}

/* Our usual callback function */
static void callback( GtkWidget *widget,
                      gpointer   data )
{
   g_print ("Hello again - %s was pressed\n", (char *) data);
}

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

```
GtkWidget *button;
GtkWidget *box;
gtk_init (&argc, &argv);

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

gtk_window_set_title (GTK_WINDOW (window), "Pixmap'd Buttons!");

/* It's a good idea to do this for all windows. */
g_signal_connect (G_OBJECT (window), "destroy",
                  G_CALLBACK (gtk_main_quit), NULL);

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

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


/* สร้างปุ่มใหม่ */
button = gtk_button_new ();

/*กำหนด signal ให้กับปุ่ม โดยกำหนดให้เมื่อคลิ๊กที่ปุ่มโปรแกรมจะไปเรียกฟังก์ชัน  call back ที่
ชื่อ “callback” และส่งข้อความ “cool button” ไปด้วย*/
g_signal_connect (G_OBJECT (button), "clicked",
                  G_CALLBACK (callback), (gpointer) "cool button");

/* เรียกใช้ฟังก์ชันที่ทำหน้าที่สร้าง box ขึ้นมาใหม่ แล้วใส่รูปภาพและข้อความเข้าไปใน box */
box = xpm_label_box ("info.xpm", "cool button");
```

```
   /* Pack and show all our widgets */
   gtk_widget_show (box);

/* สั่งให้ทำการ pack box ที่ได้เข้าไปในปุ่ม */
   gtk_container_add (GTK_CONTAINER (button), box);

  /* คำสั่งแสดงปุ่ม */
   gtk_widget_show (button);

  /* pack button เข้าไปไว้ใน window */
   gtk_container_add (GTK_CONTAINER (window), button);

   gtk_widget_show (window);

   /* Rest in gtk_main and wait for the fun to begin! */
   gtk_main ();

   return 0;
}
```

นอกจาก clicked แล้วยังมี signal อื่นที่เป็นของปุ่ม ดังนี้

- pressed – จะส่ง signal ก็ต่อเมื่อปุ่มถูกกด
- released – จะส่ง signal ก็ต่อเมื่อปุ่มถูกปล่อย
- clicked - จะส่ง signal ก็ต่อเมื่อปุ่มถูกกดและปล่อย
- enter - จะส่ง signal ก็ต่อเมื่อปุ่มถูก enter
- leave - จะส่ง signal ก็ต่อเมื่อเลื่อนเมาท์ออกจากปุ่ม

3.2 Toggle Buttons

ปุ่มแบบ toggle ก็มีลักษณะคล้ายกับปุ่มแบบทั่วไป ยกเว้นเพียงปุ่มแบบ toggle จะมีสถานะอยู่ 2 สถานะถ้าจะเป็นได้เพียงหนึ่งสถานะเท่านั้นซึ่งการจะเปลี่ยนสถานะทำได้โดยการคลิ๊กหรือกดที่ปุ่ม เมื่อกดที่ปุ่มก็จะเป็นสถานะใหม่และเมื่อคลิ๊กอีกครั้งก็จะเปลี่ยนสถานะเป็นสถานะเดิม

ปุ่มแบบ toggle ยังสามารถนำไปใช้กับปุ่มแบบ check หรือปุ่มแบบ radio ก็ได้ ซึ่งจะกล่าวถึงในภายหลัง

วิธีการสร้างปุ่มแบบ toggle มีดังนี้

```
GtkWidget *gtk_toggle_button_new( void );
GtkWidget *gtk_toggle_button_new_with_label( const gchar *label );
GtkWidget *gtk_toggle_button_new_with_mnemonic( const gchar *label );
```

ลักษณะการสร้างก็จะคล้ายกับการสร้างปุ่มแบบทั่วไป โดยฟังก์ชันแรกใช้สร้างปุ่ม toggle แบบเปล่า ฟังก์ชันที่ 2 จะมี label ด้วย ฟังก์ชันที่ 3 จะเป็นปุ่มที่มี label และมีเครื่องหมาย “_” อยู่ด้านหน้าเพื่อช่วยในการแยกคำ

เพื่อที่จะตรวจสอบสถานะของปุ่ม toggle ทำได้โดยตรวจสอบจาก “active” field และกำหนดฟังก์ชัน callback สำหรับsignal ของปุ่ม toggle คือ “toggled” สามารถใช้ฟังก์ชันตามตัวอย่าง

```
void toggle_button_callback (GtkWidget *widget, gpointer data)
{
  if (gtk_toggle_button_get_active (GTK_TOGGLE_BUTTON (widget)))
  {
    /* If control reaches here, the toggle button is down */
    } else {
    /* If control reaches here, the toggle button is up */
  }
}
```

ถ้าต้องการกำหนดสถานะให้กับปุ่ม toggle ทำได้โดยใช้ฟังก์ชัน

```
void gtk_toggle_button_set_active( GtkToggleButton *toggle_button,
                    gboolean      is_active );
```

argument แรกคือปุ่มที่สร้างและ argument ที่สองคือสถานะของปุ่มว่าเป็นถูก check หรือไม่ check ให้กำหนดเป็น TRUE หรือ FALSE โดยค่า default จะกำหนดเป็น FALSE

3.3 Check Buttons

ปุ่ม check นั้นจะมีลักษณะที่คล้ายกับปุ่ม toggle เพราะได้รับการถ่ายทอดคุณสมบัติมาจากปุ่ม toggle แต่จะมีความแตกต่างเล็กน้อย ปุ่ม check จะมีสี่เหลี่ยมเล็ก ๆ อยู่ด้านหน้าและมีข้อความต่อมาทางด้านขวา โดยมักใช้กำหนดสถานะปิดหรือเปิดให้กับ application ลักษณะฟังก์ชันที่ใช้สร้างคล้ายกับการสร้างปุ่มแบบทั่วไป ดังนี้

```
GtkWidget *gtk_check_button_new( void );
GtkWidget *gtk_check_button_new_with_label ( const gchar *label );
GtkWidget *gtk_check_button_new_with_mnemonic ( const gchar *label );
```

ฟังก์ชันที่ใช้ตรวจสอบสถานะของปุ่มให้ใช้ฟังก์ชันเดียวกันกับการตรวจสอบสถานะปุ่ม toggle

3.4 Radio Buttons

ปุ่ม radio ก็คล้ายกับปุ่ม check ต่างกันที่การจัดกลุ่มโดยปุ่ม radio มีเพียงปุ่มเดียวในกลุ่มเท่านั้นที่ถูก check มีประโยชน์ในการสร้าง application ที่ต้องการให้เลือก option ฟังก์ชันในการสร้างปุ่ม radio มีดังนี้

```
GtkWidget *gtk_radio_button_new( GSList *group );

GtkWidget *gtk_radio_button_new_from_widget( GtkRadioButton *group );
```

```
GtkWidget *gtk_radio_button_new_with_label( GSList *group,
                                const gchar  *label );

GtkWidget* gtk_radio_button_new_with_label_from_widget( GtkRadioButton *group,
                                const gchar    *label );

GtkWidget *gtk_radio_button_new_with_mnemonic( GSList *group,
                                const gchar  *label );

GtkWidget *gtk_radio_button_new_with_mnemonic_from_widget( GtkRadioButton *group,
                                const gchar  *label );
```

ทุกฟังก์ชันจะต้องการ argument ที่เป็น group ของ radio เพื่อจะจัดการกับหน้าที่ของ radio อย่างเหมาะสม ในครั้งแรกที่เรียกใช้ฟังก์ชัน gtk_radio_button_new() หรือ gtk_radio_button_new_with_label() จะผ่านค่า argument ตัวแรกเป็น NULL และในการสร้างกลุ่มของ radio จะใช้ฟังก์ชัน

```
GSList *gtk_radio_button_get_group( GtkRadioButton *radio_button );
```

สิ่งที่สำคัญอย่างหนึ่งคือ ฟังก์ชัน GSList *gtk_radio_button_get_group(); จะต้องใช้ในการสร้างปุ่มใหม่ทุกครั้ง เพื่อจะผ่านไปเป็น argument แรกของการสร้างปุ่ม ผลที่ได้จะผ่านไปเป็น argument ของการสร้างปุ่มต่อไปเรื่อย ๆ ดังแสดงตัวอย่างได้ดังนี้

```
button2 = gtk_radio_button_new_with_label(
            gtk_radio_button_get_group (GTK_RADIO_BUTTON (button1)),
            "button2");
```

หรือใช้ในอีกรูปแบบ ดังนี้

```
button2 = gtk_radio_button_new_with_label_from_widget(
                GTK_RADIO_BUTTON (button1),
            "button2");
```

ถ้าต้องการกำหนดปุ่มที่ถูก check เป็นค่า default ทำได้ดังนี้

```
void gtk_toggle_button_set_active( GtkToggleButton *toggle_button,
                        gboolean        state );
```

ซึ่งลักษณะการใช้งานคล้ายกับปุ่ม toggle สามารถเปลี่ยนสถานะการ chek ของปุ่ม radio โดยการเลือกคลิ๊กที่ช่อง radio ที่ต้องการ

ตัวอย่างการใช้งานปุ่ม radio มีดังนี้

```
void gtk_toggle_button_set_active( GtkToggleButton *toggle_button,
                        gboolean        state );
#include <glib.h>
#include <gtk/gtk.h>

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

int main( int   argc,
        char *argv[] )
{
   GtkWidget *window = NULL;
   GtkWidget *box1;
   GtkWidget *box2;
   GtkWidget *button;
```

```
GtkWidget *separator;
GSList *group;

gtk_init (&argc, &argv);

window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

g_signal_connect (G_OBJECT (window), "delete_event",
                  G_CALLBACK (close_application),
        NULL);

gtk_window_set_title (GTK_WINDOW (window), "radio buttons");
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);

/* สร้างปุ่ม radio ที่มี label */
button = gtk_radio_button_new_with_label (NULL, "button1");
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);

/* จัดกลุ่มให้กับปุ่ม radio */
group = gtk_radio_button_get_group (GTK_RADIO_BUTTON (button));
```

```
button = gtk_radio_button_new_with_label (group, "button2");

/* กำหนดให้ check  ปุ่ม radio */
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button), TRUE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);

/* สร้างปุ่มใหม่พร้อมกับการกำหนดกลุ่ม */
button = gtk_radio_button_new_with_label_from_widget (GTK_RADIO_BUTTON (button),
                                                      "button3");
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);

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");
g_signal_connect_swapped (G_OBJECT (button), "clicked",
                    G_CALLBACK (close_application),
                    G_OBJECT (window));
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;
}
```