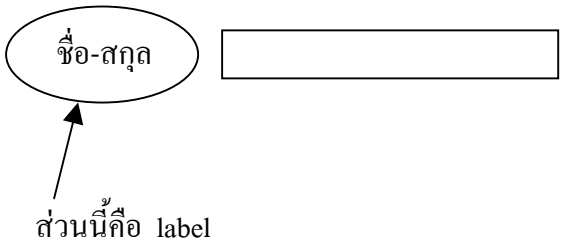
# บทที่ 4

## Widget เบ็ดเตล็ด (Miscellaneous widget)

### 4.1 Label widget

Label จากชื่อก็สามารถเข้าใจได้ว่า widget ชนิดนี้เอาไว้เพื่อเป็นใส่ข้อความ เหมือนกับป้ายต่างๆที่เป็นตัวอธิบาย เช่น




คำสั่งต่างๆที่เกี่ยวข้องกับการสร้าง label มีดังต่อไปนี้

```
GtkWidget *gtk_label_new( const char *str );
```

เป็นคำสั่งเพื่อเอาไว้สร้าง label ใหม่ขึ้นมา

```
Void gtk_label_set_justify( GtkLabel        *label,
                            GtkJustification  jtype );
```

เป็นคำสั่งเอาไว้เพื่อจัดการการจัดอักษรของ label โดยที่ **jtype** เราสามารถใส่ค่าต่างๆต่อไปนี้ได้

**GTK_JUSTIFY_LEFT**

**GTK_JUSTIFY_RIGHT**

**GTK_JUSTIFY_CENTER ( เป็นค่าเริ่มต้นของตัว widget label)**

**GTK_JUSTIFY_FILL**

**Void gtk_label_set_line_wrap (GtkLabel *label,**
**gboolean wrap);**

เป็นคำสั่งที่เอาไว้ตัดคำโดยอัตโนมัติ ถ้าเรากำหนดที่ **wrap** เป็น true

**ตัวอย่างโปรแกรม**

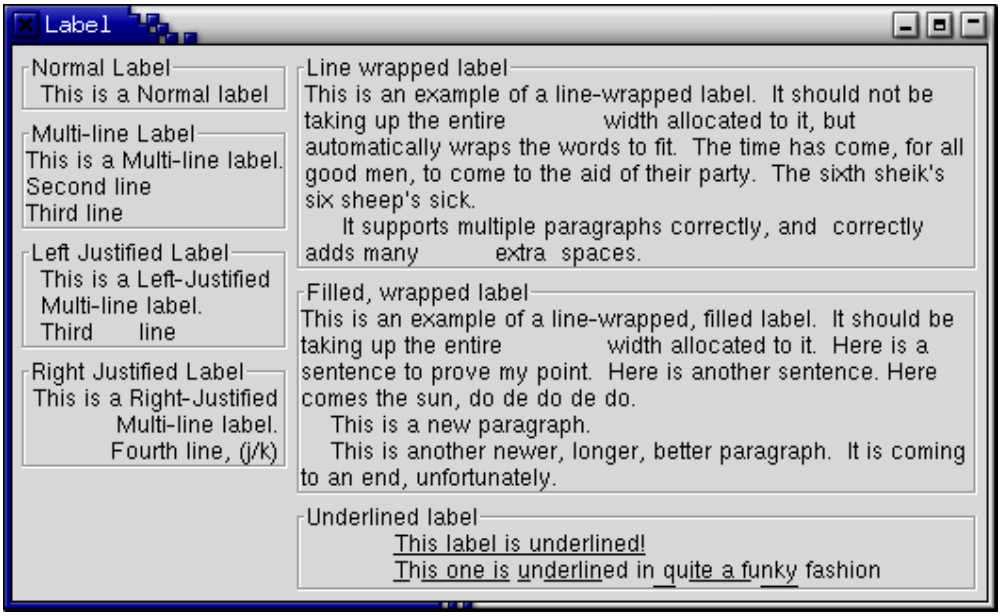
```
#include <gtk/gtk.h>
int main( int   argc,
        char *argv[] )
{
  static GtkWidget *window = NULL;
  GtkWidget *hbox;
  GtkWidget *vbox;
  GtkWidget *frame;
  GtkWidget *label;

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

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

  gtk_window_set_title (GTK_WINDOW (window), "Label");
  vbox = gtk_vbox_new (FALSE, 5);
  hbox = gtk_hbox_new (FALSE, 5);
  gtk_container_add (GTK_CONTAINER (window), hbox);
  gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
```

```
gtk_container_set_border_width (GTK_CONTAINER (window), 5);

frame = gtk_frame_new ("Normal Label");
label = gtk_label_new ("This is a Normal label");
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

frame = gtk_frame_new ("Multi-line Label");
label = gtk_label_new ("This is a Multi-line label.\nSecond line\n" \
                       "Third line");
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

frame = gtk_frame_new ("Left Justified Label");
label = gtk_label_new ("This is a Left-Justified\n" \
                       "Multi-line label.\nThird      line");
gtk_label_set_justify (GTK_LABEL (label), GTK_JUSTIFY_LEFT);
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

frame = gtk_frame_new ("Right Justified Label");
label = gtk_label_new ("This is a Right-Justified\nMulti-line label.\n" \
                       "Fourth line, (j/k)");
gtk_label_set_justify (GTK_LABEL (label), GTK_JUSTIFY_RIGHT);
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

vbox = gtk_vbox_new (FALSE, 5);
gtk_box_pack_start (GTK_BOX (hbox), vbox, FALSE, FALSE, 0);
frame = gtk_frame_new ("Line wrapped label");
```

```
label = gtk_label_new ("This is an example of a line-wrapped label.  It " \
                       "should not be taking up the entire            " /* big space to test
spacing */\
                       "width allocated to it, but automatically " \
                       "wraps the words to fit.  " \
                       "The time has come, for all good men, to come to " \
                       "the aid of their party.  " \
                       "The sixth sheik's six sheep's sick.\n" \
                       "     It supports multiple paragraphs correctly, " \
                       "and  correctly   adds "\
                       "many         extra  spaces. ");
gtk_label_set_line_wrap (GTK_LABEL (label), TRUE);
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

frame = gtk_frame_new ("Filled, wrapped label");
label = gtk_label_new ("This is an example of a line-wrapped, filled label.  " \
                       "It should be taking "\
                       "up the entire            width allocated to it.  " \
                       "Here is a sentence to prove "\
                       "my point.  Here is another sentence. "\
                       "Here comes the sun, do de do de do.\n"\
                       "    This is a new paragraph.\n"\
                       "    This is another newer, longer, better " \
                       "paragraph.  It is coming to an end, "\
                       "unfortunately.");
gtk_label_set_justify (GTK_LABEL (label), GTK_JUSTIFY_FILL);
gtk_label_set_line_wrap (GTK_LABEL (label), TRUE);
gtk_container_add (GTK_CONTAINER (frame), label);
gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);
```

```
  frame = gtk_frame_new ("Underlined label");
  label = gtk_label_new ("This label is underlined!\n"
                         "This one is underlined in quite a funky fashion");
  gtk_label_set_justify (GTK_LABEL (label), GTK_JUSTIFY_LEFT);
  gtk_label_set_pattern (GTK_LABEL (label),
"_____ _ _____ _ _____   __ _____ ___");
  gtk_container_add (GTK_CONTAINER (frame), label);
  gtk_box_pack_start (GTK_BOX (vbox), frame, FALSE, FALSE, 0);

  gtk_widget_show_all (window);
  gtk_main ();
  return 0;
}
```

จากโปรแกรมตัวอย่างจะได้ผลลัพธ์ดังต่อไปนี้



## 4.2 The Tooltips Object

tooltips เป็นตัวที่คล้ายกับของใน window คือเมื่อเรานำเม้าส์ไปชี้ที่ปุ่มหรือ widget ใดๆก็ตามประมาณ 1-2 วินาที จะมีข้อความที่เราตั้งไว้ปรากฏขึ้นมาให้เห็น เพื่อเป็นเครื่องอำนวยความสะดวกแก่การใช้โปรแกรมที่เราเป็นผู้พัฒนาขึ้นมา โดยที่ก่อนอื่นเราจะต้องเรียกใช้คำสั่งดังต่อไปนี้

```
GtkTooltips *gtk_tooltips_new( void );
```

เมื่อเราได้สร้าง tooltip ขึ้นมาแล้วเราต้องทำการตั้งค่าให้กับ tooltip ด้วยคำสั่งดังต่อไปนี้

```
void gtk_tooltips_set_tip( GtkTooltips *tooltips,
                           GtkWidget   *widget,
                           const gchar *tip_text,
                           const gchar *tip_private );
```

**ตัวอย่างการใช้งาน**

```
GtkTooltips *tooltips;
GtkWidget *button;

tooltips = gtk_tooltips_new ();
button = gtk_button_new_with_label ("button 1");

gtk_tooltips_set_tip (tooltips, button, "This is button 1", NULL);
```

## 4.3 Dialogs

Dialogs เป็น widget ที่เป็นพื้นฐานของการสร้าง window ซึ่งจะใช้มากเมื่อเวลาที่เราต้องการให้มีหน้าต่าง popup ขึ้นมา Dialog widget เป็น widget ที่ไม่มีความซับซ้อนเลยก็ว่าได้ เมื่อเปรียบเทียบกับ widget แบบอื่น ๆ เวลาที่ถูกเรียก ให้ทำงาน widget แบบนี้ก็จะเป็น window ที่มีองค์ประกอบไม่กี่อย่างภายในหน้าจอ เพื่อให้เราใช้ตามจุดประสงค์ หรือความต้องการของเรา โดยมาก

จะพบว่า 1 dialog จะเป็น window ที่จะทำงานเรื่องใดเรื่องหนึ่งเพียงอย่างเดียว โดยที่ตัว Dialogs จะมีโครงสร้างข้อมูลดังต่อไปนี้

```
struct GtkDialog
{
    GtkWindow window;
    GtkWidget *vbox;
    GtkWidget *action_area;
};
```

จากโครงสร้างดังกล่าว หมายความว่า เราสร้าง window ขึ้นมาอันหนึ่ง แล้วมี pack box แบบ vbox อยู่ด้านบน หลังจากนั้นจะมีส่วนคั่นต่อจาก vbox และจะมี "action_area" เป็น hbox คำสั่งที่ใช้สร้าง dialog คือ

```
GtkWidget * gtk_dialog_new(void);
```

ซึ่งการสร้าง dialog ขึ้นมานั้นจะเหมือนกับการสร้าง window หนึ่งขึ้นมาแต่เป็น window ที่มีโครงสร้างข้อมูลให้ใช้เท่านั้นหากเราต้องการใส่ label กับปุ่มลงไปสามารถทำได้ดังนี้

```
button = gtk_button_new_with_label("....");
gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window)->action_area),
                   button, TRUE, TRUE, 0);
gtk_widget_show(button);
```

***** การใส่ปุ่ม *****

```
label = gtk_label_new(".........");
gtk_box_pack_start(GTK_BOX(GTK_DIALOG(window)->vbox),
                   label, TRUE, TRUE, 0);
gtk_widget_show(label);
```

*****การใส่ label*****

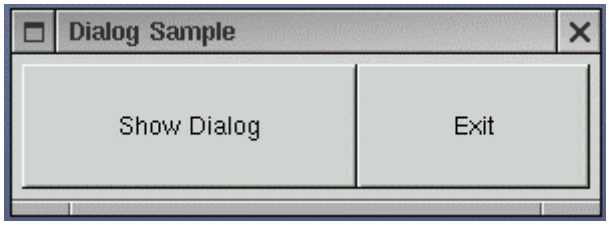
**ตัวอย่างโปรแกรม**

```
#include <gtk/gtk.h>
void hideDlg(GtkWidget *widget, gpointer data)
{
   gtk_widget_hide((GtkWidget*)data);
}
void callDialog(GtkWidget *widget, gpointer data)
{
  GtkWidget * dlg;
  GtkWidget * label;
  GtkWidget * button;

  dlg = gtk_dialog_new();
  button = gtk_button_new_with_label("OK");
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dlg)->action_area),
                button, TRUE, TRUE, 0);
  gtk_signal_connect(GTK_OBJECT(button), "clicked",
                GTK_SIGNAL_FUNC(hideDlg), dlg);
  gtk_widget_show(button);

  label = gtk_label_new("Oh.. WOW  now Dialog showned!");
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(dlg)->vbox),
                label, TRUE, TRUE, 0);
  gtk_widget_show(label);

  gtk_widget_show(dlg);
}
int main(int argc, char *argv[])
{
```

```
GtkWidget  * window;
GtkWidget  * box;
GtkWidget  * button;

gtk_init(&argc, &argv);

window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
gtk_window_set_title(GTK_WINDOW(window),"Dialog Sample");
gtk_container_border_width(GTK_CONTAINER(window),5);
gtk_signal_connect(GTK_OBJECT(window), "delete_event",
            GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

box = gtk_hbox_new(FALSE, 0);
gtk_container_add(GTK_CONTAINER(window), box);

button = gtk_button_new_with_label("Show Dialog");
gtk_signal_connect(GTK_OBJECT(button), "clicked",
            GTK_SIGNAL_FUNC(callDialog), NULL);
gtk_box_pack_start(GTK_BOX(box), button, TRUE, TRUE, 0);
gtk_widget_show(button);

button = gtk_button_new_with_label("Exit");
gtk_signal_connect(GTK_OBJECT(button), "clicked",
            GTK_SIGNAL_FUNC(gtk_main_quit), NULL);
gtk_box_pack_start(GTK_BOX(box),button,TRUE,TRUE,0);
gtk_widget_show(button);

gtk_widget_show(box);
gtk_widget_show(window);
gtk_main();
```

```
    return(0);
}
```

จากโปรแกรมตัวอย่างจะได้ผลลัพธ์ดังต่อไปนี้




## 4.4 Text Enties

คุณสมบัติพิเศษของ widget แบบ text-entries มีดังนี้

- สามารถรับและแสดงข้อความที่พิมพ์ใน text box แบบ 1 บรรทัด
- สามารถกำหนดจำนวนตัวอักษรสูงสุดได้
- สามารถเรียกฟังก์ชันพิเศษในการปรับปรุงข้อความได้ ดังนี้

1. การทับคำเดิม (replace)
2. กำหนดค่าตำแหน่งของ cursor ได้
3. อ่านข้อความจาก text ได้
4. กำหนดให้เพิ่ม text ได้
5. กำหนดให้อ่านอย่างเดียวได้ (ไม่สามารถแก้ไขข้อความได้)
6. กำหนดให้แสดง/ซ่อนข้อความได้
7. เลือกแสดงเป็นช่วงได้ เช่น ตั้งแต่ตัวที่ 6 ถึง 12 เป็นต้น

รูปแบบของคำสั่ง จะเป็นดังนี้

**GtkWidget *gtk_entry_new(void);** *** สร้าง text entry ตามปกติ ***

**GtkWidget *gtk_entry_new_with_max_length(guint16 max); ;**

*** สร้าง text entry โดยกำหนดความยาวของ text entry ด้วย ***

ทั้ง 2 คำสั่งเป็นการสร้าง text entry

**void gtk_entry_set_text (GtkExtry *entry,**

**const gchar *text);**

*** เป็นการกำหนดค่าข้อความให้กับ entry ***

**void gtk_entry_append_text (GtkEntry *entry,**

**const gchar *text);**

*** เป็นการกำหนดค่าข้อความให้กับ entry โดยต่อท้ายข้อความเก่า***

**void gtk_entry_prepend_text (GtkEntry *entry,**

**const gchar *text);**

*** เป็นการกำหนดค่าข้อความให้กับ entry โดยแทรกข้อความใหม่หน้าข้อความเก่า***

**void gtk_entry_set_position (GtkEntry *entry,**

**gint position);**

*** เป็นการตั้งค่าของ cursor ว่าจะอยู่ตำแหน่งไหนของข้อความใน entry ***

**gchar * gtk_entry_get_text(GtkEntry *entry);**

*** เป็นการอ่านข้อความจาก entry ***

**void gtk_entry_set_editable (GtkEntry *entry,**

**gboolean editable);**

*** กำหนดให้สามารถแก้ไขหรือไม่สามารถแก้ไขข้อความใน entry ได้ ***

**void gtk_entry_set_visibility (GtkEntry *entry,**

**gboolean visible);**

*** กำหนดให้แสดงอักษร หรือ ไม่แสดงเป็นตัวอักษรได้ ***

```
void gtk_entry_select_region (Gtkentry *entry,
                              gint start,
                              gint end);
*** กำหนดว่าจะเลือกข้อความใน entry ตั้งแต่ส่วนไหนถึงไหนได้ ***
```

**ตัวอย่างโปรแกรม**

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

void enter_callback( GtkWidget *widget,
                     GtkWidget *entry )
{
  const gchar *entry_text;
  entry_text = gtk_entry_get_text (GTK_ENTRY (entry));
  printf("Entry contents: %s\n", entry_text);
}

void entry_toggle_editable( GtkWidget *checkbutton,
                            GtkWidget *entry )
{
  gtk_editable_set_editable (GTK_EDITABLE (entry),
                             GTK_TOGGLE_BUTTON (checkbutton)->active);
}

void entry_toggle_visibility( GtkWidget *checkbutton,
                              GtkWidget *entry )
{
```

```
  gtk_entry_set_visibility (GTK_ENTRY (entry),
                                GTK_TOGGLE_BUTTON (checkbutton)->active);
}

int main( int   argc,
        char *argv[] )
{
   GtkWidget *window;
   GtkWidget *vbox, *hbox;
   GtkWidget *entry;
   GtkWidget *button;
   GtkWidget *check;
   gint tmp_pos;

   gtk_init (&argc, &argv);

   /* create a new window */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_widget_set_size_request (GTK_WIDGET (window), 200, 100);
   gtk_window_set_title (GTK_WINDOW (window), "GTK Entry");
   g_signal_connect (G_OBJECT (window), "destroy",
                        G_CALLBACK (gtk_main_quit), NULL);
   g_signal_connect_swapped (G_OBJECT (window), "delete_event",
                                G_CALLBACK (gtk_widget_destroy),
                                G_OBJECT (window));

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

```
entry = gtk_entry_new ();
gtk_entry_set_max_length (GTK_ENTRY (entry), 50);
g_signal_connect (G_OBJECT (entry), "activate",
                  G_CALLBACK (enter_callback),
                  (gpointer) entry);
gtk_entry_set_text (GTK_ENTRY (entry), "hello");
tmp_pos = GTK_ENTRY (entry)->text_length;
gtk_editable_insert_text (GTK_EDITABLE (entry), " world", -1, &tmp_pos);
gtk_editable_select_region (GTK_EDITABLE (entry),
                            0, GTK_ENTRY (entry)->text_length);
gtk_box_pack_start (GTK_BOX (vbox), entry, TRUE, TRUE, 0);
gtk_widget_show (entry);

hbox = gtk_hbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (vbox), hbox);
gtk_widget_show (hbox);
check = gtk_check_button_new_with_label ("Editable");
gtk_box_pack_start (GTK_BOX (hbox), check, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (check), "toggled",
                  G_CALLBACK (entry_toggle_editable), (gpointer) entry);
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (check), TRUE);
gtk_widget_show (check);

check = gtk_check_button_new_with_label ("Visible");
gtk_box_pack_start (GTK_BOX (hbox), check, TRUE, TRUE, 0);
g_signal_connect (G_OBJECT (check), "toggled",
                  G_CALLBACK (entry_toggle_visibility), (gpointer) entry);
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (check), TRUE);
gtk_widget_show (check);
```

```
  button = gtk_button_new_from_stock (GTK_STOCK_CLOSE);
  g_signal_connect_swapped (G_OBJECT (button), "clicked",
                            G_CALLBACK (gtk_widget_destroy),
                            G_OBJECT (window));
  gtk_box_pack_start (GTK_BOX (vbox), 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;
}
```

จากโปรแกรมตัวอย่างจะได้ผลลัพธ์ดังต่อไปนี้

## 4.5 Combo Box

combo box จะคล้ายกับเป็น list ที่เราเคยใช้ในการทำ web ซึ่งเมื่อเรากดที่ลูกศรจะมีให้เราเลือกข้อมูลได้ว่าเราต้องการข้อมูลอะไร ลักษณะของ combo box จะมีการใส่ข้อความเป็น list เรียงตามกันไปโดยเราไม่ต้องยุ่งยากในเรื่องของการประกาศขนาดของ list หรือต้องจองพื้นที่ให้กับ list ซึ่งตัว gtk เองจะเป็นตัวจัดการให้ทั้งหมด combo box จะมี structure ดังต่อไปนี้

```
struct _GtkCombo {
                    GtkHBox hbox;
                    GtkWidget *entry;
                    GtkWidget *button;
                    GtkWidget *popup;
                    GtkWidget *popwin;
                    GtkWidget *list;        };
```

คำสั่งต่างๆของ combo box ได้แก่

**GtkWidget *gtk_combo_new( void );**
*** เป็นการสร้าง combo box ขึ้นมาใหม่ ***

**void gtk_combo_set_popdown_strings ( GtkCombo *combo,**
**GList *strings );**
*** ทำให้ combo box สามารถแสดงเป็น list ออกมาให้เห็นได้ ***

**GList *g_list_append ( GList *glist,**
**gpointer data );**
*** เป็นการแทรกข้อมูลต่างๆลงไปใน list ***

**void gtk_combo_set_use_arrows ( GtkCombo *combo,**
**gboolean val );**
*** กำหนดให้ใช้เป็นcombo box แบบมีลูกศรหรือไม่ก็ได้ ***

**ตัวอย่างการใช้งาน**

```
GtkWidget       *combo;
GList           *glist = NULL;
```

```
combo = gtk_combo_new( void );
glist = g_list_append (glist, "String 1");
glist = g_list_append (glist, "String 2");
glist = g_list_append (glist, "String 3");
 glist = g_list_append (glist, "String 4");
gtk_combo_set_popdown_strings ( GTK_COMBO (combo), glist );

gtk_combo_set_use_arrows( GTK_COMBO (combo), TRUE  );

gtk_entry_set_text ( GTK_ENTRY (GTK_COMBO (combo)->entry), "My String." );
```

## 4.6 Calendar

Calendar เป็น widget ที่แสดงเป็นรูปของปฏิทินซึ่งเราจะนำไปใช้กันมากในการทำโปรแกรมต่างๆ ปฏิทินที่ได้มาจาก widget นี้จะเป็นปฏิทินที่จัดการกับวันที่ๆเป็นวันที่พิเศษให้แล้ว เช่น ปีที่เดือนกุมภาพันธ์มี 29 วัน ผู้ใช้ไม่ต้องทำอะไรกับตัวปฏิทินเลย ทั้งนี่เรายังสามารถดึงวันที่ปัจจุบันมาแสดงได้อีกด้วยคำสั่งของการสร้างปฏิทินมีดังต่อไปนี้

**GtkWidget *gtk_calendar_new( void );**
*** เป็นคำสั่งที่ใช้สร้างปฏิทินขึ้นมา ***

```
void gtk_calendar_get_date ( GtkCalendar *calendar,
                             guint      *year,
                             guint      *month,
                             guint      *day );
```
*** เป็นคำสั่งที่ใช้รับค่าวันที่ปัจจุบันมา ***

```
void gtk_calendar_display_options ( GtkCalendar  *calendar,
                                    GtkCalendarDisplayOptions  flags );
```
*** เป็นคำสั่งที่ใช้กำหนดว่าจะให้แสดงปฏิทินในรูปแบบใด ***
เราสามารถกำหนดค่า flags ได้ดังต่อไปนี้

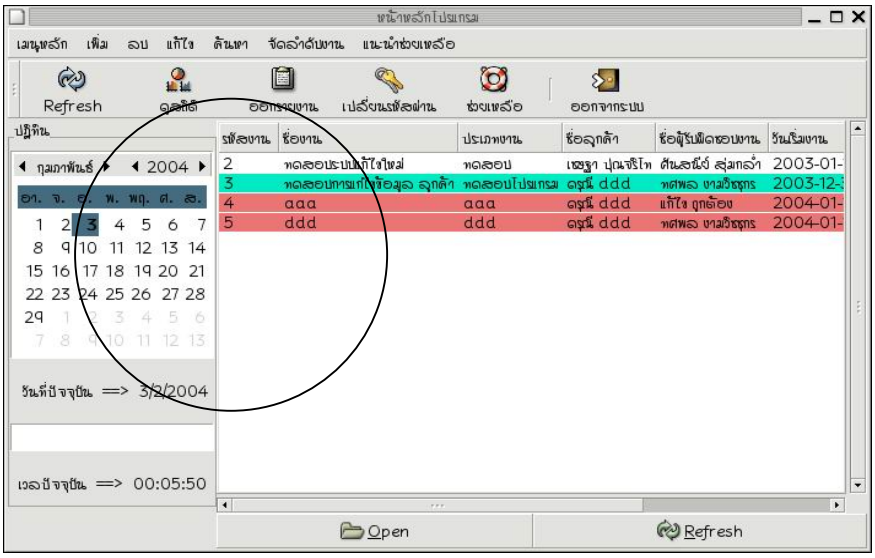
**GTK_CALENDAR_SHOW_HEADING**

**GTK_CALENDAR_SHOW_DAY_NAMES**

**GTK_CALENDAR_NO_MONTH_CHANGE**

**GTK_CALENDAR_SHOW_WEEK_NUMBERS**

**GTK_CALENDAR_WEEK_START_MONDAY**




รูปแบบปฏิทินที่ได้เมื่อใช้คำสั่งสร้างปฏิทินโดยไม่กำหนด flags ใดๆ