(1条消息)如何使用GTK让系统显示托盘图标 - hushiwei1993的专栏 - CSDN博客

使用下面的演示代码时，注意图标的路径，否则不能显示托盘图标。如果编译不成功，请注意编译命令中的字符是否是英文的字符。

介绍：

这篇文章解释了GtkStatusIcon部件的使用，这个GtkStatusIcon部件用来在系统托盘里放置图标。为了和用户交互提示条和弹出菜单可以加入到这个图标里。这个图标也可以闪烁。

如何在系统托盘里放置图标：

下列函数可以用来创建一个新的状态图标：

```
    GtkStatusIcon* gtk_status_icon_new ();
```

GtkStatusIcon* gtk_status_icon_new_from_pixbuf   (GdkPixbuf *pixbuf);

GtkStatusIcon* gtk_status_icon_new_from_file   (const gchar *filename);

GtkStatusIcon* gtk_status_icon_new_from_stock   (const gchar *stock_id);

GtkStatusIcon* gtk_status_icon_new_from_icon_name (const gchar *icon_name);

为了设置一个提示条文本可以用下面的函数：
void gtk_status_icon_set_tooltip (GtkStatusIcon *icon, const gchar *tooltip_text);

为了闪烁一个图标用下面的函数：
void gtk_status_icon_set_blinking (GtkStatusIcon *icon, gboolean blinking);

当用户激活这个托盘图标，托盘图标发送"activate"信号；当用户指出一个菜单应该显示时，托盘图标发送一个"popup-menu"信号。

演示程序：
下面的程序当用户按下最小化按钮时，创建一个托盘图标，隐藏应用窗口。使用托盘图标的弹出菜单应用窗口变得有效。当和托盘图标交互时信号被处

理。

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

static void trayView(GtkMenuItem *item, gpointer user_data);
static void trayExit(GtkMenuItem *item, gpointer user_data);
static void trayIconActivated(GObject *trayIcon, gpointer data);
static void trayIconPopup(GtkStatusIcon *status_icon, guint button, guint32
activate_time, gpointer popUpMenu);
static void destroy (GtkWidget*, gpointer);
static gboolean delete_event (GtkWidget*, GdkEvent*, gpointer);
static gboolean window_state_event (GtkWidget *widget, GdkEventWindowState
*event, gpointer user_data);

int main(int argc, char *argv[])
{
    gtk_init (&argc, &argv);

    GtkWidget *window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "GtkStatusIcon Example");
    gtk_widget_set_size_request (window, 200, -1);
    //set try icon file
    GtkStatusIcon *trayIcon  = gtk_status_icon_new_from_file ("/root/Desktop
/icon.png");
    //set popup menu for tray icon
    GtkWidget *menu, *menuItemView, *menuItemExit;
    menu = gtk_menu_new();
    menuItemView = gtk_menu_item_new_with_label ("View");
    menuItemExit = gtk_menu_item_new_with_label ("Exit");
    g_signal_connect (G_OBJECT (menuItemView), "activate", G_CALLBACK
(trayView), window);
    g_signal_connect (G_OBJECT (menuItemExit), "activate", G_CALLBACK
(trayExit), NULL);
    gtk_menu_shell_append (GTK_MENU_SHELL (menu), menuItemView);
    gtk_menu_shell_append (GTK_MENU_SHELL (menu), menuItemExit);
    gtk_widget_show_all (menu);
    //set tooltip
    gtk_status_icon_set_tooltip (trayIcon, "MsgWatcherGTK");
    //connect handlers for mouse events
    g_signal_connect(GTK_STATUS_ICON (trayIcon), "activate",
GTK_SIGNAL_FUNC (trayIconActivated), window);
    g_signal_connect(GTK_STATUS_ICON (trayIcon), "popup-menu",
GTK_SIGNAL_FUNC (trayIconPopup), menu);
```

```
    gtk_status_icon_set_visible(trayIcon, FALSE); //set icon initially invisible


    GtkWidget *menuBar, *menuItemTopLvl, *mainMenu, *mainMenuItemExit;
    menuBar = gtk_menu_bar_new ();
    menuItemTopLvl = gtk_menu_item_new_with_label ("Menu");
    gtk_menu_shell_append (GTK_MENU_SHELL (menuBar), menuItemTopLvl);

    mainMenu = gtk_menu_new ();
    gtk_menu_item_set_submenu (GTK_MENU_ITEM (menuItemTopLvl),
mainMenu);
    mainMenuItemExit = gtk_menu_item_new_with_label ("Quit");
    g_signal_connect (G_OBJECT (mainMenuItemExit), "activate",
G_CALLBACK (trayExit), NULL);
    gtk_menu_shell_append (GTK_MENU_SHELL (mainMenu),
mainMenuItemExit);

    g_signal_connect (G_OBJECT (window), "destroy", G_CALLBACK (destroy),
NULL);
    g_signal_connect (G_OBJECT (window), "delete_event", G_CALLBACK
(delete_event), trayIcon);
    g_signal_connect (G_OBJECT (window), "window-state-event",
G_CALLBACK (window_state_event), trayIcon);
    gtk_container_add (GTK_CONTAINER (window), menuBar);
    gtk_widget_show_all (window);
    gtk_main ();
    return 0;
}

static void trayView(GtkMenuItem *item, gpointer window)
{
    gtk_widget_show(GTK_WIDGET(window));
    gtk_window_deiconify(GTK_WINDOW(window));
}

static void trayExit(GtkMenuItem *item, gpointer user_data)
{
    printf("exit");
    gtk_main_quit();
}

static void trayIconActivated(GObject *trayIcon, gpointer window)
{
    gtk_widget_show(GTK_WIDGET(window));
    gtk_window_deiconify(GTK_WINDOW(window));
}

static void trayIconPopup(GtkStatusIcon *status_icon, guint button, guint32
activate_time, gpointer popUpMenu)
{
    gtk_menu_popup(GTK_MENU(popUpMenu), NULL, NULL,
gtk_status_icon_position_menu, status_icon, button, activate_time);
}

static void destroy (GtkWidget *window, gpointer data)
{
  gtk_main_quit ();
}

static gboolean delete_event (GtkWidget *window, GdkEvent *event, gpointer
data)
{
    return FALSE;
}

static gboolean window_state_event (GtkWidget *widget, GdkEventWindowState
*event, gpointer trayIcon)
{
   if(event->changed_mask == GDK_WINDOW_STATE_ICONIFIED &&
(event->new_window_state == GDK_WINDOW_STATE_ICONIFIED ||
event->new_window_state == (GDK_WINDOW_STATE_ICONIFIED |
GDK_WINDOW_STATE_MAXIMIZED)))
   {
      gtk_widget_hide (GTK_WIDGET(widget));
      gtk_status_icon_set_visible(GTK_STATUS_ICON(trayIcon), TRUE);
   }
   else if(event->changed_mask == GDK_WINDOW_STATE_WITHDRAWN
&& (event->new_window_state == GDK_WINDOW_STATE_ICONIFIED ||
event->new_window_state == (GDK_WINDOW_STATE_ICONIFIED |
GDK_WINDOW_STATE_MAXIMIZED)))
   {
      gtk_status_icon_set_visible(GTK_STATUS_ICON(trayIcon), FALSE);
   }
   return TRUE;
}
```

上面的代码可以用下面的命令编译：

```
gcc trayicon.c -o trayicon `pkg-config --cflags --libs gtk+-2.0`.
```

License