Introducción a la programación en el Entorno GNOME

Claudio Saavedra

csaavedra@gnome.org,
http://www.gnome.org/~csaavedra/

Resumen Se presentan algunos de los conceptos necesarios para desarrollar una aplicación que se integre adecuadamente en el escritorio GNOME. Se presentan conceptos elementales de GTK+ y Glade junto a libglade; el uso del área de notificación, del administrador de archivos recientes, y también de las notificaciones flotantes.

1. Introducción

El presente artículo presenta algunos de los conceptos elementales al desarrollo de aplicaciones que se integren en el escritorio GNOME. Para esto, presentaremos las tecnologías existentes actualmente en la plataforma de GNOME, y aquellas que son cercanas y facilitan la tarea de los programadores.

La integración en GNOME parte por la estética, por lo que usaremos el toolkit de GTK+ y la biblioteca libglade para generar interfaces. Además, es importante la integración con el resto del escritorio, por lo que mostraremos una pequeña parte de las muchas herramientas con las que se cuenta para ésta tarea, incluyendo el área de notificación, el administrador de archivos recientes y las notificaciones flotantes.

2. Introducción a GTK+

GTK+ es la biblioteca para el desarrollo de interfaces gráficas de GNOME. A partir de ésta, podemos crear desde simples aplicaciones por medio de los widgets del stock, hasta complejas interfaces que utilicen nuestros widgets personalizados.

La biblioteca GTK+ está escrita originalmente en el lenguaje C, y está sostenida por un conjunto de bibliotecas que incluyen GLib, ATK, cairo, y libxml. Pese a ser una biblioteca completamente desarrollada en C, el sistema GObject le brinda toda la potencia necesaria para otorgar orientación a objetos a sus desarrolladores.

2.1. Ejemplo introductorio

El siguiente es un ejemplo básico de un programa desarrollado con GTK+. No es más que una ventana con un botón y una etiqueta de texto en su interior, como en la figura 1.

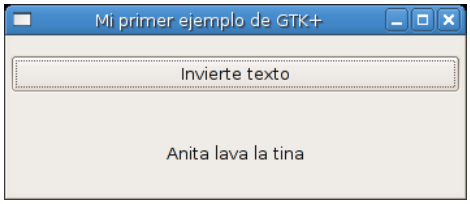

Figura 1. Simple ejemplo de una aplicación escrita con GTK+.

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

void
on_button_clicked (GtkButton *button,
                   gpointer data)
{
    g_print ("boo!\n");
}

GtkWidget *
construct_window (void)
{
    GtkWidget *window;
    GtkWidget *vbox;
    GtkWidget *button;
    GtkWidget *label;
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window),
                          "Mi primer ejemplo de GTK+");

    vbox = gtk_vbox_new (TRUE, 0);

    button = gtk_button_new_with_label ("Invierte texto");

    label = gtk_label_new ("Anita lava la tina");

    gtk_box_pack_start (GTK_BOX (vbox), button, FALSE, FALSE, 0);
    gtk_box_pack_start (GTK_BOX (vbox), label, FALSE, FALSE, 0);

    gtk_container_add (GTK_CONTAINER (window), vbox);

    g_signal_connect (G_OBJECT (button),
                      "clicked",
                      G_CALLBACK (on_button_clicked),
                      label);
```

```
    return window;
}

int
main (gint argc, gchar **argv)
{
    gtk_init (&argc, &argv);
    GtkWidget *window = construct_window ();

    gtk_window_set_default_size (GTK_WINDOW (window), 400, 300);
    gtk_widget_show_all (window);
    g_signal_connect (G_OBJECT (window), "delete-event",
                      gtk_main_quit, NULL);

    gtk_main ();

    return 0;
}
```

Antes de continuar, es importante saber que para poder compilar software desarrollado con GTK+, se utiliza la herramienta `pkg-config`. Ésta nos permite simplificar los parámetros que debemos pasar a nuestro compilador al momento de compilar, ya que se preocupa de enlazar correctamente con la ubicación de las bibliotecas y cabeceras en nuestro sistema:

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

Debe notarse que el carácter ` es un acento grave, como en la palabra *Viquipèdia*.

En el ejemplo vemos los elementos básicos que comúnmente identificaremos en una interfaz desarrollada con GTK+. Dentro de la función `main`, está comúnmente la inicialización del sistema GTK+, la creación de un objeto `window` que representa la ventana de nuestra aplicación. Luego definimos el valor de una propiedad de nuestro objeto `window`, le mostramos, en conjunto con todos sus elementos hijos, y conectamos una señal a él.

2.2. Señales

Como podemos ver, hay dos conceptos elementales en esta función que debemos comprender: el uso de señales (*"delete-event"*, en este caso), y el uso de los métodos de las clases de GObject (el método `GtkWindow::set_default_size` y `GtkWidget::show_all`).

Una señal permite que nuestro programa ejecute una cierta tarea cuando nuestros objetos cambien de estado o cuando se produzca una interacción del usuario con la UI. En particular, tenemos *"delete-event"*, una señal que pertenece a todos los widgets de GTK+, y que se emite cada vez que el objeto es eliminado. En nuestro ejemplo, usamos esta señal en el objeto `window` para que, al eliminar

la ventana, se llame a la función `gtk_main_quit`, que se encargará de finalizar el ciclo de GTK+, y por tanto, finalizar nuestra aplicación.

Otra señal que utilizamos en nuestro ejemplo, es la señal ‘‘`clicked`’’, que se emite cada vez que un usuario pulsa sobre un botón. En general, para conectar una función `function` a una señal ‘‘`signal`’’, en un objeto `object`, podemos utilizar la función `g_signal_connect` de la siguiente manera:

```
g_signal_connect (object, "signal", G_CALLBACK (function), data);
```

donde `data` representa alguna variable que contenga datos que sea de nuestro interés facilitar a `function`.

2.3. Contenedores

Otro concepto importante de GTK+ es el uso de *contenedores*. Un contenedor es un widget que sirve para contener a otros widgets. Algunos son bastante sencillos y permiten contener a sólo un widget hijo (las ventanas, por ejemplo), y otros son más complejos y permiten anidar un número no determinado de hijos en un ordenamiento particular (`GtkVBox` para ordenamientos verticales, `GtkHBox` para ordenamientos horizontales, `GtkTable` para ordenamientos tabulares).

En la función `create_window` vemos un caso particular del uso de contenedores. *GtkVBox* es usado para alinear un botón con una caja de texto. Una vez que contamos con un objeto `vbox`, es simple usar un método como `GtkBox::pack_start` para empaquetar el botón y la caja de texto en él.

2.4. Además

GTK+ tiene muchísimos controles completos que permiten desarrollar aplicaciones tanto sencillas como complejas. Sería difícil resumirlos todos en un sólo artículo, pero algunas de las más frecuentemente usadas son:

`GtkButton:` Un widget que representa un botón en una aplicación.
`GtkLabel:` Permite desplegar una etiqueta de texto en la aplicación.
`GtkTreeView:` Permite desplegar un árbol o una lista de datos complejos.
`GtkEntry:` Una entrada de texto de una línea.
`GtkTextView:` Un completo control para visualizar texto multilínea y con formato.

Para una introducción breve a GTK+, recomendamos la lectura de [7]. Además, el *Libro GNOME* [3] es un recurso bastante completo para entender los conceptos básicos de GTK+, pese a que está un poco desactualizado.

3. Glade y libglade

3.1. Glade-3

Glade es una aplicación que nos permite diseñar interfaces de manera interactiva. Con ella, la generación de interfaces complejas se simplifica bastante.

En sus inicios, Glade nos permitía generar el código en C necesario para generar las interfaces que creamos con ella. Sin embargo, esta práctica tiene muchas desventajas, en especial con respecto a la mantención del código generado, por lo que ya no se recomienda. Por esto, James Henstridge escribió la biblioteca libglade, que a partir del archivo XML que define una interfaz creada con glade es capaz de generarla en tiempo de ejecución, sin la necesidad de código en C generado. Desde entonces, ya no es recomendable usar el código generado por glade, y se da preferencia al uso de esta biblioteca.

Por esto, la versión más reciente de Glade, ya no incluye la generación de código, y sólo permite el uso de los archivos XML de glade en conjunto con libglade. Además, la próxima versión de GTK+ incluirá un módulo llamado GtkBuilder, que brinda las mismas facilidades que libglade para la generación automática de interfaces a partir de XML.

En la figura 2 vemos la interfaz principal de Glade mientras creamos una interfaz sencilla.

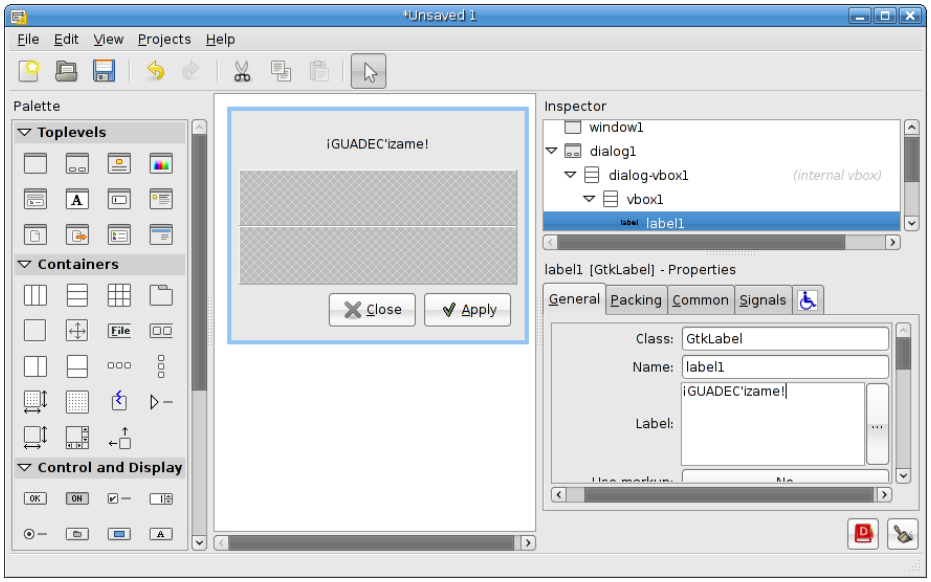

Figura 2. Glade-3, el generador de interfaces para GNOME.

3.2. libglade

Una vez que hayamos finalizado nuestra interfaz, podemos cargarla desde una aplicación, por medio de libglade. Esto es tan simple como hacer

```
GladeXml *xml;
GtkWidget *window;

xml = glade_xml_new (archivo.glade, "main", NULL);
glade_xml_signal_autoconnect (xml);
window = glade_xml_get_widget (xml_window_main, "main");

gtk_widget_show_all (window);
```

Con esto, habremos cargado, desde un archivo de Glade llamado `archivo.glade`, una ventana cuya especificación se encuentra etiquetada como `main` en el XML. Además, mediante el método `GladeXML::signal_autoconnect` es posible conectar de manera automática todas las señales y retrollamadas que hubiéramos definido por medio de Glade-3.

Contando con el puntero a nuestra ventana, ahora podemos manipularla como cualquier otro widget de GTK+. Por esto, podemos usar el método `GtkWindow::show_all` sobre el widget que hemos cargado desde glade.

Para compilar una aplicación que incluya la biblioteca libglade, debemos utilizar la cabecera correspondiente, es decir

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

Además, la línea de compilación debe incluir una llamada a `pkg-config` que incluya como parámetro a la biblioteca libglade:

```
$ gcc -o ejemplo ejemplo.c ‘pkg-config --cflags --libs gtk+-2.0 libglade-2.0‘
```

3.3. Otros recursos sobre glade y libglade

Una guía para el uso de la biblioteca libglade para facilitar la creación de la interfaz en conjunto con autotools es [4]. Ahí se puede revisar un ejemplo simple pero completo sobre el uso de esta biblioteca.

4. GTK+ e Integración con el escritorio

A continuación, describiremos algunos de los elementos que GTK+ brinda para integrar aplicaciones con el escritorio GNOME. Estos nos permiten darle a nuestras aplicaciones un toque profesional y acabado.

4.1. Archivos recientes

Desde GTK+ 2.10, la biblioteca brinda soporte integrado para el manejo de archivos utilizados recientemente. Con este, podemos crear un menú de archivos recientes para nuestra aplicación o si lo preferimos un selector de archivos recientes. Además, los archivos manejados por el administrador de archivos recientes serán agregados automáticamente al menú de archivos recientes en la barra de GNOME.

El modo más sencillo de utilizar el administrador de archivos recientes consiste en obtener una instancia del administrador, y agregar la URI del archivo que deseemos agregar a la lista de archivos usados recientemente. Por ejemplo, en nuestro método para abrir archivos podemos usar

```
abrir_archivo (gchar *file)
{
    GtkRecentManager *manager;

    manager = gtk_recent_manager_get_default ();
    gtk_recent_manager_add_item (manager, file_uri);

    /* código para abrir el archivo */

}
```

con esto, hemos agregado al archivo referenciado por `file_uri` a la lista de archivos recientemente utilizados. Ahora bien, podemos crear un diálogo con la lista de archivos usados recientemente. Para esto, todo lo que necesitamos es usar el widget `GtkRecentChooserDialog`:

```
GtkWidget *dialog;

dialog = gtk_recent_chooser_dialog_new ("Archivos Recientes",
                                        parent_window,
                                        GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
                                        GTK_STOCK_OPEN, GTK_RESPONSE_ACCEPT,
                                        NULL);

if (gtk_dialog_run (GTK_DIALOG (dialog)) == GTK_RESPONSE_ACCEPT)
  {
    GtkRecentInfo *info;

    info = gtk_recent_chooser_get_current_item (GTK_RECENT_CHOOSER (dialog));
    open_file (gtk_recent_info_get_uri (info));
    gtk_recent_info_unref (info);
  }

gtk_widget_destroy (dialog);
```

Esto nos presentará un diálogo que permitirá al usuario seleccionar un archivo de la lista, tal como el de la figura 3.

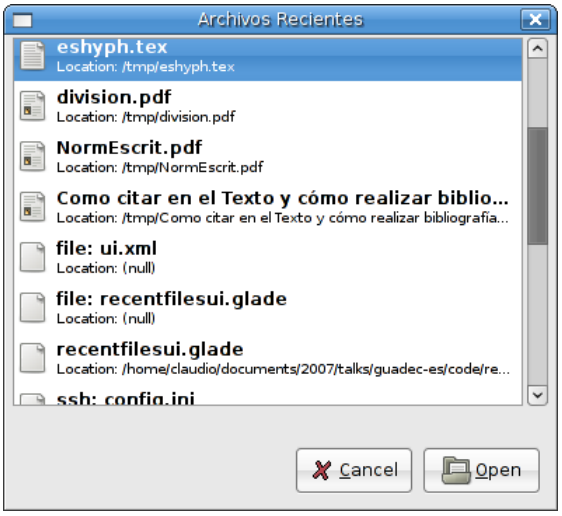

Figura 3. Diálogo de archivos recientes de GTK+.

4.2. Área de notificación

Un icono en el área de notificación permite a nuestra aplicación informar al usuario de ciertos eventos que requieran su atención. Dependiendo de la plataforma donde nuestra aplicación esté ejecutándose, existirá un *área de notificación*, donde estos iconos se despliegan.

Con GTK+, el código para mostrar un icono en el área de notificación es bastante simple. Supongamos que deseamos mostrar un icono para notificar al usuario de una advertencia en nuestro programa. Entonces, tan solo necesitamos el siguiente código:

```
GtkStatusIcon *icon;

icon = gtk_status_icon_new_from_stock (GTK_STOCK_DIALOG_WARNING);
gtk_status_icon_set_tooltip  (icon, "Esta es una advertencia...");
gtk_status_icon_set_visible (icon, TRUE);
```

El resultado obtenido con este código sería como el de la figura 4.

5. Notificaciones

Además del área de notificación, es posible usar la biblioteca *libnotify* para notificar al usuario mediante un mensajes flotantes. Esta biblioteca requiere que

Figura 4. Icono de notificación en la barra de GNOME.

además esté ejecutándose el demonio *notification-daemon*, ya que es éste el que presenta las notificaciones y se encarga de administrarlas. Este demonio es usado ampliamente en un escritorio GNOME moderno, lo que nos permite asegurar que las notificaciones serán útiles para la mayoría de nuestros usuarios.

Para mostrar una notificación sencilla, sólo necesitamos inicializar *libnotify* llamando a la función `notify_init ()`. Esta llamada debe realizarse antes que cualquier otra llamada a la API de *libnotify*. Entonces, cuando queramos notificar al usuario de algún suceso, simplemente creamos una nueva notificación, y la mostramos:

```
NotifyNotification *n;

notify_init("Basics");

n = notify_notification_new ("Atención",
                             "Esta notificación no tiene ninguna utilidad, "
                             "además de mostrar el uso de libnotify",
                             NULL, NULL);

notify_notification_set_timeout (n, 3000);
notify_notification_show (n, NULL);
g_object_unref (G_OBJECT (n));
```

Al ejecutarse, se mostrará en pantalla una notificación sencilla, como la de la figura 5.

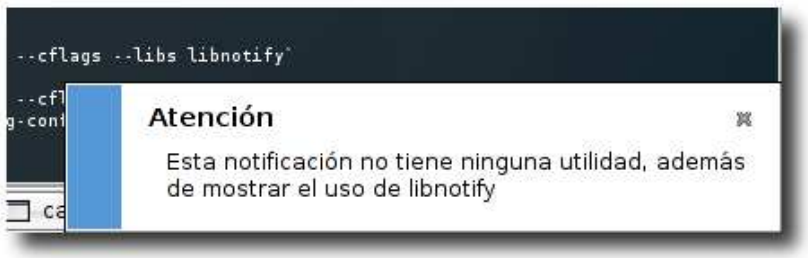

Figura 5. Una notificación cortesía de libnotify.

Pero *libnotify* permite crear notificaciones más complejas, que incluyan también, iconos, imágenes, y otros widgets que permitan hacer notificaciones más interesantes. Sugerimos entonces que se revisen los ejemplos incluidos con el código de la biblioteca.

6. Otros recursos

Recomendamos siempre revisar tanto la API de las bibliotecas que hemos mencionados, como los ejemplos incluídos con el código de ellas. Además, el tutorial de GTK+ [2] es una buena introducción a muchos de los conceptos más complejos. Para tener una idea de la estética deseada en las aplicaciones para GNOME, es importante consultar la Guía de Interfaces de Usuario de GNOME [8].

Referencias

1. GTK+ developers: *GTK+ Reference Manual*. `http://developer.gnome.org/doc/API/2.0/gtk/`.
2. Gale, T., Main, I., and the GTK+ team: *GTK+ 2.0 Tutorial*. `http://www.gtk.org/tutorial/`.
3. Moya, R., et al.: *Programación en el entorno GNOME*. `http://libros.es.gnome.org/librognome/librognome/librognome/book1.html`.
4. Saavedra, C.: *Iniciando un proyecto GNOME con libglade y autotools*. `http://www.gnome.org/~csaavedra/documents/usinglibglade/index.html`.
5. Warkus, M.: *The Official GNOME 2 Developer's Guide*.
6. Newren, E.: *Developing with Gnome In C, C++, Perl, and Python*. `http://www.gnome.org/~newren/tutorials/developing-with-gnome/html/`.
7. Saavedra, C.: *Widgets, Controladores y Señales: Desarrollando Aplicaciones con GTK+*. en proceedings de *GUADEC-es 2006, Vilanova y la Geltrú*, España. Junio 2006.
8. The GNOME Usability Project: *GNOME Human Interface Guidelines 2.0*. `http://developer.gnome.org/projects/gup/hig/2.0/`