Programación en GNOME.

Miguel de Icazan
miguel@gnu.org

Instituto de Ciencias Nucleares
UNAM.

La biblioteca glib

- Estructuras de datos abstractas:
  - Listas
  - Arboles
  - Tablas de hash genéricas
  - Arreglos dinámicos
- Utilerías de portabilidad
- Wrapper de memoria con soporte para debugging.

Glib: main loop

- Un main loop general:
  - Timers
  - Monitoreo de fuentes de datos
    - Archivos/descriptores de archivos (entrada, salida).
    - Otras fuentes de archivos pueden añadirse: GIOSource
  - controlador de tiempo ocioso.
- Usado en ORBit, el toolkit GTK+ y el servidor de calendario

Manejo de recursos

- Conteo de referencias.
  - Usado inicialmente en GTK+.
  - Uso extendido a muchas partes del sistema.
- OBJETO_ref, OBJETO_unref.
- Estructuras de datos en glib
- Recursos del sistema de ventanas en Gdk:
  - Ventanas
  - Fuentes
  - Imágenes
- Objetos de GTK (Gtk).

Enfoque en software correcto

Uso de precondiciones.

- Puntos de entrada públicos:
  - g_return_if_fail
  - g_return_val_if_fail
- Uso de g_assert
- g_assert_not_reached
- g_warning, g_warning_only_once, g_error

La clase base de GTK

```
struct _GtkObject {
  GtkObjectClass *klass;
  guint32 flags;
  guint ref_count;
  GData *object_data;
};
struct _GtkObjectClass {
  GtkType type;
  guint *signals;
  guint nsignals;
  guint n_args;
  GSList *construct_args;
  void (*set_arg) (GtkObject *object, GtkArg *arg, guint arg_id);
  void (*get_arg) (GtkObject *object, GtkArg *arg, guint arg_id);
  void (*shutdown) (GtkObject *object);
  void (*destroy)  (GtkObject *object);
  void (*finalize) (GtkObject *object);
};
```

El sistema de objetos GTK

```
typedef struct {
        GtkObject padre;

        int  dato_instancia_1;
        char *dato_instancia_2;
} MiObjeto;

typedef struct {
        GtkObjectClass clase_padre;

        /* métodos */
        void  (*metodo_1) (MiObjeto *objeto, int arg1, char *arg2);
        FILE *(*metodo_2) (MiObjeto *objeto);

        /* senales */
        void *(*senal_1)  (MiObjeto *objeto);
} MiObjetoClass;
```

El sistema de objetos GTK

Registrado de tipos:

```
GtkType
mi_objeto_get_type (void)
{
        static GtkType type = 0;

        if (!type){
                GtkTypeInfo info = {
                        "MiObjeto",
                        sizeof (MiObjeto),
                        sizeof (MiObjetoClass),
                        (GtkClassInitFunc) mi_objeto_class_init,
                        (GtkObjectInitFunc) mi_objeto_instance_init,
                };
                type = gtk_type_unique (gtk_object_get_type (), &info);
        }
        return type;
}
```

El sistema de objetos GTK

```
static void
mi_objeto_class_init (MiObjetoClass *clase)
{
        GtkObjectClass *object_class = (GtkObjectClass *) class;
        mi_objeto_parent_class = gtk_type_class (gtk_object_get_type ());

        gnome_object_signals [SENAL_1] =
                gtk_signal_new ("senal-1", GTK_RUN_LAST,
                         object_class->type,
                               GTK_SIGNAL_OFFSET(MiObjeto,senal_1),
                               gtk_marshal_NONE__NONE,
                               GTK_TYPE_NONE, 0);
        gtk_object_class_add_signals (object_class, mi_objeto_signals,
                                 LAST_SIGNAL);
        object_class->destroy = mi_objeto_destroy;
        clase->senal_1 = mi_senal_1;
        clase->metodo_1 = mi_metodo_1;
        clase->metodo_2 = mi_metodo_2;
}
```

El sistema de objetos GTK

- Verificación de tipos.

```
#define GNOME_OBJECT_TYPE          (gnome_object_get_type ())
#define GNOME_OBJECT(o)             (GTK_CHECK_CAST ((o),
                                 GNOME_OBJECT_TYPE, GnomeObject))
#define GNOME_OBJECT_CLASS(k)      (GTK_CHECK_CLASS_CAST((k),
                                 GNOME_OBJECT_TYPE,
                                 GnomeObjectClass))
#define GNOME_IS_OBJECT(o)         (GTK_CHECK_TYPE ((o),
                                GNOME_OBJECT_TYPE))
#define GNOME_IS_OBJECT_CLASS(k) (GTK_CHECK_CLASS_TYPE ((k),
                                GNOME_OBJECT_TYPE))
```

El sistema de objetos GTK

Ejemplo del uso de la verificación de tipos

```
void
mi_objeto_op_1 (MiObjeto *objeto, char *datos)
{
        g_return_if_fail (objeto != NULL);
        g_return_if_fail (MI_IS_OBJETO (objeto));
        g_return_if_fail (datos != NULL);

        objeto->dato_instancia_2 = g_strdup (datos);
}
```

Conección de señales

Uso:

```
gtk_signal_connect (objeto, "senal", GTK_SIGNAL_FUNC (funcion), datos)
```

Ejemplo:

```
gtk_signal_connect (GTK_OBJECT (boton), "clicked",
                    GTK_SIGNAL_FUNC (function), datos)
```

GTK+ el toolkit

- Gdk
  - La capa de bajo nivel para acceder al sistema de ventanas.
- Gtk
  - Componentes gráficos y objetos para programación de aplicaciones gráficas.
  - GtkWidget, objeto base para el GUI.

Gdk_Imlib

Extensión para manipulación de imágenes

- Cargar y guardar imágenes
  - Módulos nativos para los formatos populares.
  - ImageMagik y netpbm último recurso
- Manipulación de imágenes.
  - Rotar (0, 90, 180, 270)
  - Escalar.
  - Dithering en 8 bits.
  - Usa memoria compartida.
  - Control de color.
  - Máscaras (imágenes con forma).

Gdk_Imlib

- Crear imágen (archivo, datos en línea).
  - Produce GdkImlib

```
GdkImlib *gdk_imlib_load_image (const char *nombre);
```

- Configuración de la imágen en cualquier punto
  - Altera la imágen GdkImlib.
- Generación de el Pixmap y el Bitmap
  - Caché de imágenes/pixmaps/bitmaps incluido.

```
gdk_imlib_render (GdkImlib *imagen, int width, int height);
```

Gdk_Imlib -- parte 2

- Copiar/Referenciar el GdkPixmap/GdkBitmap
  - Produce un Bitmap/Pixmap usable por Gdk/Gtk.

```
GdkPixmap     *gdk_imlib_copy_image(GdkImlibImage * image);
GdkBitmap     *gdk_imlib_copy_mask(GdkImlibImage * image);
GdkPixmap     *gdk_imlib_move_image(GdkImlibImage * image);
GdkBitmap     *gdk_imlib_move_mask(GdkImlibImage * image);
```

- Destruir la imágen:

```
void          *gdk_imlib_destroy_image(GdkImlibImage * image);
```

Biblioteca gnome
Rutinas de acceso a datos de configuración.

- Rutinas de la forma:
  gnome_config_get_XXXXX (const char *llave_y_omision)
  gnome_config_set_XXXXX (const car *llave, XXX valor)
- Formato de la llave:
  - En espacio de GNOME:
    /archivo/seccion/llave=valor_por_omision
  - Referencia absoluta:
    =/archivo=/seccion/llave=valor_por_omision
- Tiene que ser movido a un servidor CORBA

Biblioteca gnome
Tipos MIME.

- Usados para clasificar el contenido de los archivos.
- Por patrón del nombre de archivo (gnome-mime)
  - Las reglas se instalan en:
    - $prefix/mime-info/*.mime
    - ~/.gnome/mime-info/*mime
- Por contenido (gnome-magic).
  - Este es particularmente caro.
  - Usa una base de datos binaria (compilada).

Biblioteca gnome
Tipos MIME

- Asociación de llaves/valores por tipo (gnome-mime-info)
- Algunos significados predefinidos:
  - open
  - view
  - edit
  - icon-filename
- Las reglas se instalan en:
  - $prefix/mime-info/*.keys
  - ~/.gnome/mime-info/*.keys

Metadatos en GNOME

- Asocian información arbitraria a un archivo por llave.
  - íconos en línea (inline-png-image)
  - Posición
  - Comando para lanzar
  - Comando para aventar objetos
- Actualmente usa una base de datos db.
- En el futuro será movido a un servidor CORBA independiente.

Procesamiento de lineas de comandos

- Uso de popt para esto.
  - Generación de ayuda automática
  - Análisis de opciones correcto.
- Infraestructura para registrar parsers.
  - varias librerías registran argumentos: gtk, gnomeui, gnorba, zvt, ORBit.
- El último parser es el provisto por la aplicación.

Biblioteca gnomeui

- Implementa varios widgets compuestos.
- Iconos/Etiquetas Stock
- Creación de Menús y Barras de herramientas
  - gnome-app-helper.
- Manejo de sesiones
- Canvas de GNOME

Biblioteca gnomeui
Widgets compuestos

- Calculadora
- Entrada:
  - números.
  - nombres de archivos.
  - íconos.
  - imágenes.
  - fuentes.
- Selector de colores.
- Lista de íconos.

Biblioteca gnomeui
Nuevos widgets

- GnomeDialog
  - Clase base para diálogos de Gnome
  - GnomeMessageBox
- GnomeDock
  - Para hacer menus y toolbars flotantes.
- Zvt:
  - Emulación terminal xterm
- GtkXmHTML:
  - Motor de HTML

Biblioteca gnomeui
Manejo de sesiones

- Crear un controlador de sesión
  - gnome_master_client ();
  - conectar a las senales:
    - "save_yourself"
    - "die"
- Inicialización:
  - gnome_config_push_prefix (...get_config_prefix ());
  - ...
  - ... gnome_config_get ();
  - gnome_config_pop_prefix ();

Biblioteca gnomeui
Iconos Stock

- Iconos estándar para las aplicaciones:
  - gnome_stock_button (GNOME_STOCK_BUTTON_...)
  - gnome_stock_menu_item (tipo, texto)
- Integrado directamente con gnome-app-helper

El Canvas de GNOME

Creación:

```
canvas = gnome_canvas_new ();
canvas = gnome_canvas_new_aa ();
root = gnome_canvas_root (canvas);
```

Objetos:

```
tipo = gnome_canvas_line_get_type ();
points = gnome_canvas_points_new (2);
points->coords [0] = x0;
points->coords [1] = y0;
points->coords [2] = x1;
points->coords [3] = y1;
linea = gtk_canvas_item_new (canvas, tipo,
                     "points", points,
                     "fill_color", "black"
                     "width_units", 4.0,
                     NULL);
gnome_canvas_points_unref (points);
```

El canvas de GNOME

Conección a eventos.

```
gtk_signal_connect (linea, "event", mi_evento, NULL);

int mi_evento (GnomeCanvasItem *item, GdkEvent *event, void *data){
    if (event->type == GDK_BUTTON_PRESS){
        gnome_canvas_item_set (linea, "fill_color", "blue", NULL);
        gnome_canvas_item_grab (linea);
    }
    if (event->type == GDK_BUTTON_RELEASE){
        gnome_canvas_item_set (linea, "fill_color", "black", NULL);
        gnome_canvas_item_ungrab (linea);
    }
    return TRUE;
}
```

Biblioteca libgnorba

- Simplifica y estandariza el uso de CORBA en GNOME:
  - Seguridad:
    - Autentificación
    - Protege de ataques maliciosos.
  - Activación de servicios.
  - Servicio de nombres para aplicaciones GNOME.

Internacionalización en GNOME.

- Usamos GNU gettext.
- Aplicaciones usan gettext
- Librerías usan dgettext
- Ambos usan:
  - _(texto)

Internacionalización: Ejemplos

- Versión convencional:

```
char *dos_palabras [] = { "imp", "resionante" }

printf ("Hola");
printf (a [0]);
```

- Versión internacionalizada:

```
char *dos_palabras [] = { N_("imp"), N_("resionante") }

printf (_("Hola"));
printf (_(a [0]));
```

Documentación

- Usamos Docbook para el markup de la documentación.
- gtk-doc para extraer la documentación de los comentarios en linea:

Formato

```
/**
 * gnome_canvas_item_grab:
 * @item: The item to grab.
 * @event_mask: Mask of events that will be sent to this item.
 * @cursor: If non-NULL, the cursor that will be used while ...
 * @etime: The timestamp required for grabbing the mouse, or ...
 *
 * Specifies that all events that match the specified event ...
 * item, and also grabs the mouse by calling gdk_pointer_grab()
 * grabbing the pointer.  If @cursor is not NULL, then that ...
 * The @etime parameter is the timestamp required for grabbing...
 *
 * Return value: If an item was already grabbed, it returns ...
 * item is not visible, then it returns %GrabNotViewable.  Else, ...
 * calling gdk_pointer_grab().
 **/
```