

Conferencia:

Programación gráfica en lenguaje xBase en GNU/Linux

Audiencia: Desarrolladores (especialmente en lenguajes tales como CA-Clipper)

Duración: 1 hora incluidas las preguntas.

Temario:

- ¿Cuál lenguaje xBase elegimos?

En el mundo del SL tenemos varios:

1. Harbour
2. xHarbour
3. Clip

¿Porqué Clip?

- Antecedentes de la necesidad de la programación gráfica en lenguajes xBase

Durante el año 2004 y 2005 hubo varias discusiones en el grupo de clip-castellano acerca de la “programación gráfica” Crear con Clip GUIs aprovechando nuestro conocimiento del lenguaje.

Luego de algunos intercambios de correos electrónicos se llegó a la conclusión de que lo mejor era usar la biblioteca de widgets GTK2. La ausencia de una IDE en Clip hizo natural el querer aprovechar a Glade para “pintar” la interfaz y luego crear a partir de ella las aplicaciones.

Sergio Zayas creó un programa llamado: glade2clip que tomaba lo creado por Glade y armaba un programa Clip con las llamadas adecuadas a las funciones de Clip.

Pronto se vio que esto no era un camino muy prometedor y Sergio creó la biblioteca clip-glade2 que hace con Clip lo que hace la libglade con C o C++ uno no necesita crear el código en estos lenguajes a partir del archivo XML creado con glade2, sino que “conecta” esta interfaz con los eventos programados en Clip que son invocados mediante señales y mágicamente la interfaz es pintada en la pantalla y los eventos administrados con una llamada a una función.

- Problemas que se suscitan ante el desafío de la programación gráfica

Clip es un lenguaje extremadamente poderoso y flexible, que tiene las siguientes formas para hacer interfaces gráficas:

1. GTK
2. GTK2
3. Fivewin implementada con GTK
4. Clip-UI
5. Clip-glade2

Nos concentraremos en la forma de hacer una agenda muy simple mediante clip-glade2 por las siguientes razones:

1. Es sencillo crear las interfaces gráficas mediante Glade2
2. Es lo único que entiendo por ahora de la interfaz gráfica en GNU/Linux :-)

3. No conozco Fivewin
4. Clip-UI carece de generador de interfaces gráficas
5. Programar directamente usando las funciones de GTK y GTK2 se ve como tedioso

- ¿Porqué GTK2?

GTK se considera obsoleta y se detuvo su desarrollo.

GTK2 es multiplataforma

Hay abundante documentación

Se puede usar Glade2

- ¿Porqué Glade?

Glade permite diseñar con facilidad las interfases

Es ubicua a cualquier distribución y funciona aun en MS-Windows

Con libglade es lo que usaremos para crear la interfaz gráfica.

- ¿Es multiplataforma?

En teoría es multiplataforma: Clip funciona con cygwin y Glade2 y GTK2 están portadas a MS-Windows.

- Distintas alternativas a GTK2 para Clip

Como mencionamos antes, está la biblioteca que emula la famosa FiveWin para Clipper.

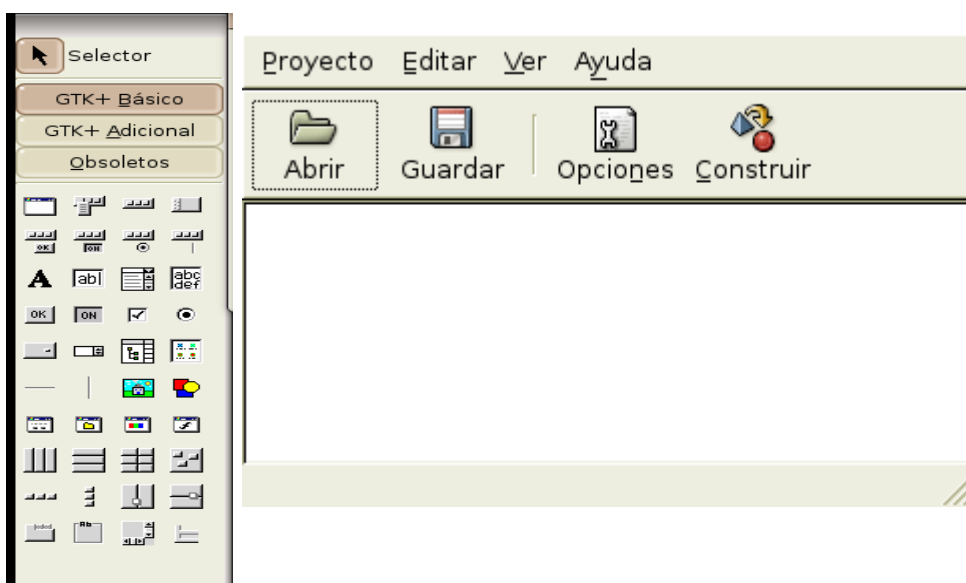
Está basada en GTK.

Clip-UI permite generar interfaces gráficas también con documentos XML, pero carece de un generador de tales documentos, tal como sería Glade2. Si se quiere ver un desarrollo hecho con Clip-UI, ver la siguiente página:

<http://eas.lrn.ru/index.php?id=screenshots>

- Primeros pasos con Glade2

Abrimos Glade2 y aparecen dos paneles:



Para el manejo de Glade y sus conceptos básicos ver la siguiente página:

<http://eddy.writelinux.com/spanish/>

Creo una ventana nueva con el icono correspondiente a la que llamamos 'window1'

- Programando una simple agenda paso por paso.

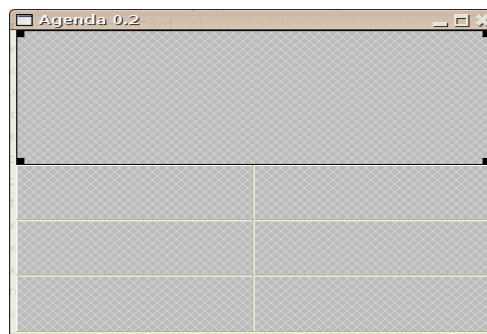
Le cambiamos el título a la ventana: “Agenda 0.2”

Si no se ven las propiedades de los widgets, es porque no está activada en el menú “Ver” de Glade.

Ahora voy a crear el contenedor para la barra de tareas, los campos y las etiquetas de los campos.

El contenedor deberá tener entonces dos renglones y al renglón de abajo lo deberé dividir a su vez en otro contenedor de tres renglones con dos columnas (los campos a mostrar serán: nombre, dirección y teléfono)

Como resultado tendremos lo siguiente:



Ahora agrego los botones que serán las acciones sobre la agenda: Agregar, Editar, Borrar, Buscar, Siguiente y Anterior.

Vemos como Glade acomoda automáticamente las dos secciones.

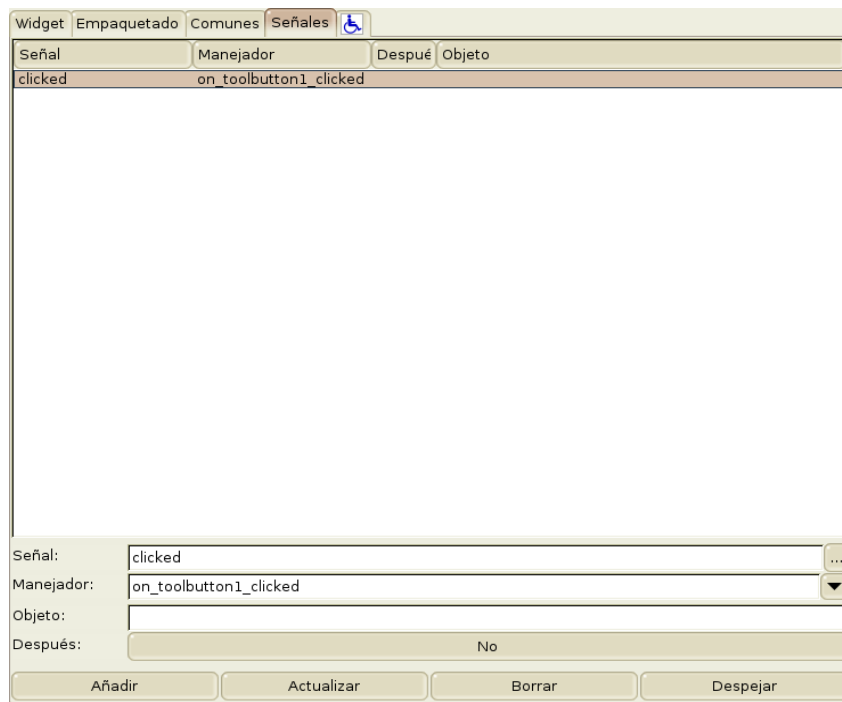
Para ello agregamos un widget “barra de herramientas” de 5 botones.

Luego vamos agregando los botones y editando sus propiedades

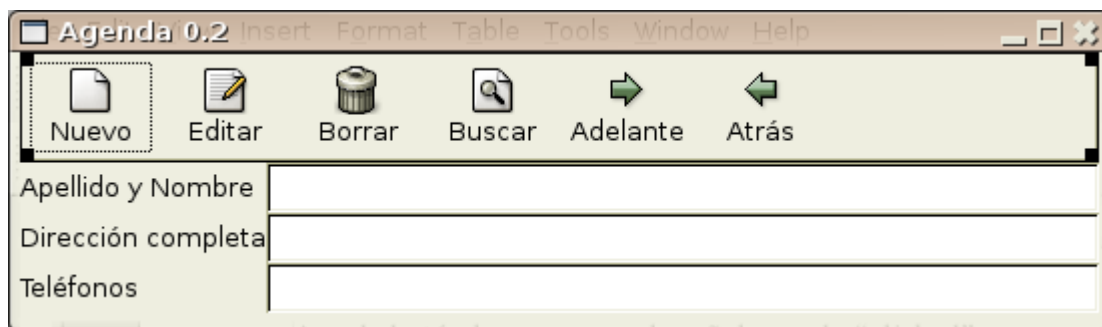
Ahora agregamos los botones a la barra de herramientas.

Recordar que para que muestre el icono con el texto incluido hay que seleccionar "Botón de inventario".

A cada botón le agregamos la señal cuando “clicked”:



Lo que obtenemos al final, rellenando los distintos contenedores con etiquetas y campos de edición es lo siguiente:



Grabamos el proyecto como 'agenda.glade'

Ahora intentaremos hacer que con Clip nos muestre esta interfaz, sólo eso.

Siguiendo el ejemplo de Sergio Zayas en `clip-prg-1-1-16-1/cliplibs/clip-glade2/examples` cargaremos la interfaz con el siguiente código:

```

/*
 agenda.prg
 Primera prueba con clip-libglade2
*/

static _xmlfic

function xmlfic()
return ( _xmlfic )

function main( glade_file)

local xml, widget
_xmlfic := iif( empty( glade_file) , "agenda.glade" ,
glade_file )

if (!file( _xmlfic ) )
? "No se encontró el archivo de interfaz: " + _xmlfic
?
quit
endif
gtk_init()
xml_w1 := glade_xml_new( _xmlfic , "window1" )
if (xml_w1 == nil )
? "*** Error de parser, archivo: " + _xmlfic
return( 1 )
endif

widget := glade_xml_get_widget( xml_w1 , "window1" )
glade_xml_signal_autoconnect( xml_w1 ) //glade autoconecta
las señales

gtk_widgetShow( widget ) //lo mostramos si no está visible
gtk_main() // esto arranca todos los eventos

return( 0 )

```

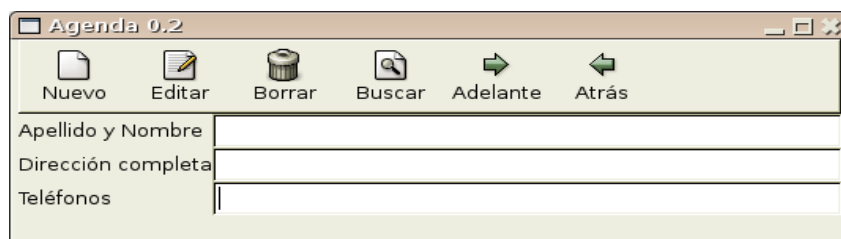
Ahora compilamos:

```
$ clip -e agenda.prg -lclip-gtk2 -lclip-glade2
```

Ejecutamos nuestra agenda:

```
./agenda
```

Vemos lo siguiente:



Hasta aquí llegamos con la primera parte de nuestra pequeña investigación.

En la próxima veremos como crear un widget para cuando picamos el botón de cerrar en la agenda.

Aprenderemos con ese ejemplo algo más de los eventos y señales en GTK2 y Glade.

Este trabajo se distribuye con la licencia “Creative Commons Share alike” tal como se menciona en:

<http://creativecommons.org/licenses/by-sa/2.0/>

- Conclusiones y preguntas.