Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Creando interfaces gráficas con Gtk2-Perl (19695 lectures)

Per **Vicenç Miralda Bover**, ***vmiralda*** (http://www.parcfluvial.org)

Creado el 19/08/2004 23:15 modificado el 19/08/2004 23:15

En este artículo veremos una pequeña introducción a la creación de GUIs a través del módulo Gtk2 de perl. Este módulo nos permitirá crear interfaces gráficas, de aspecto bastante profesional, de una manera rápida y fácil.

Índice

Con el módulo Gtk2, podemos crear GUIs de manera rápida y fácil, para programas pequeños y medianos. En primer lugar, necesitamos tener instalados los módulos Glib y Gtk2. En debian es el paquete libgtk2-perl. Para las otras distribuciones podeis bajar los 2 módulos del CPAN(6) o directamente desde la línea de comandos con *perl -MCPAN -e shell* (*install Glib* y *install Gtk2*).

Veremos una pequeña introducción a Gtk2 a través de Perl. Solamente trataré los signals (producidos por eventos), callbacks, posicionamiento de widgets, ventanas, los 7 widgets principales, menues y poco mas. Para los otros 35 widgets, *man* es vuestro amigo :-).

1.- Hello World!

Para no romper con la tradición, empezamos con el típico *Hello World!*

```
#!/usr/bin/perl -w

use Gtk2 '-init';

$ventana = Gtk2::Window->new('toplevel');
$ventana->set_title('Hello World!');
$ventana->set_border_width(20);
$boton = Gtk2::Button->new('Hello World!');
$boton->signal_connect('clicked' => sub {Gtk2->main_quit; });
```

```
$ventana->add($boton);
$ventana->show_all;

# Empieza el show...
Gtk2->main;
```

Bien, casi no necesita comentarios :-). La linea *use Gtk2 '-init'* carga el toolkit Gtk2 y lo inicializa. Sobra decir, que esta linea deberá estar presente en todos los programas :-) Con la llamada al constructor ('new') creamos una nueva instancia del objeto Window. El escalar $ventana contiene una referencia al widget que acabamos de crear.

A continuación, establecemos el título de la ventana y ponemos un borde de 20px a la ventana. Creamos un botón con el texto especificado y ahora llega lo importante. Asociamos la pulsación del boton (el evento 'clicked') a una subrutina (estrictamente, es una referencia a una subrutina). En el caso del ejemplo, estamos haciendo referencia a una subrutina anónima. Finalmente, con el método *add* añadimos el botón a la ventana creada anteriormente. Le decimos a la ventana que muestre todos los widgets que contiene y entramos en el bucle principal. A partir de aquí, el programa permanece a la espera que se produzca algún evento.

Muchas de las funciones de Gtk2 reciben como argumentos TRUE o FALSE. Si en vez de ir poniendo 1 y 0, quereis usar TRUE y FALSE, teneis que exportar las constantes TRUE y FALSE del módulo Glib al inicio de vuestros scripts. Esto es: *use Glib qw/TRUE FALSE/;*

2.- Eventos y Callbacks

Gtk es un toolkit orientado a eventos. Al llamar a *Gtk2->main* ,el programa entra en un bucle infinito y permanece a la espera de eventos. Eventos hay de muchos tipos. Un evento puede ser la pulsación de un botón por el usuario, la entrada de texto, la destrucción de un widget, etc. Cada widget tiene sus propios eventos o signals. Prácticamente todos los widgets generan alguna señal, incluso el Label.

Para que el programa haga algo, será necesario asociar los eventos a callbacks (funciones que realizaran algo en función del evento que se haya producido). Todos los widgets tienen el método *"signal_connect"* que nos permitirá asociar un evento a un callback.

```
$widget->signal_connect(evento => \&callback);
```

Asociar el click de un botón a un callback, es tan simple como:

```
$boton->signal_connect(clicked => \&haz_algo);
```

El callback no es mas que una referencia a una función (\&nombre_funcion).

```
sub haz_algo {
        my ($widget, @argumentos) = @_;
        ....
}
```

$widget contiene una referencia al widget que ha generado el evento. Obviamente, se pueden pasar argumentos a la función.

```
$boton->signal_connect(clicked => \&haz_algo, 'un_argumento');
```

El primer elemento del array @_ siempre será una referencia al widget que ha generado el evento. El resto de elementos del array @_ son los argumentos con los cuales se ha llamado a la función.

Existe una jerarquia de widgets (con *man Gtk2::index* podeis ver un listado de todos los widgets). Los widgets inferiores heredan los signals de sus widgets padre. Comentar que con *$widget->set('propiedad' => valor)* podemos cambiar qualquier propiedad de un widget (para ver las propiedades de un widget, *man Gtk2::Widget*). Por ejemplo:

```
$label->set('label' => 'Nuevo texto del Label');
```

Cambiamos la propiedad 'label' (es el texto que muestra el Label) del widget $label.

3.- Posicionando los widgets

En el toolkit Gtk2, existen dos métodos para posicionar los widgets dentro un widget contenedor (como ejemplos de Widgets contenedores podemos citar Gtk2::Window, Gtk2::VBox, Gtk2::HBox... pero hay muchos mas, incluso un botón puede ser un contenedor).
El primer método consiste en usar los widgets HBox (es una caja horizontal transparente) y VBox (la correspondiente caja vertical transparente). En la caja horizontal, por defecto, los widgets se insertan de izquierda a derecha. En la caja vertical, por defecto, los widgets se insertan de arriba a abajo.

```
# El argumento homegeneidad (booleano) controla si los widgets que
# contenga la caja tendran el mismo tamaño. El segundo argumento es
# el espacio añadido entre los objetos.
$caja = Gtk2::HBox->new( $homegeneidad, $spacing);
$caja_vertical = Gtk2::VBox->new($homegeneidad, $spacing);

# Añadimos dos botones a nuestra caja horizontal( $expand es en valor booleano que
# determina si el widget se expendirá para ocupar toda la celda(TRUE) o será la celda
# quien se adaptará al tamaño del widget(FALSE), $fill  debe ser un valor booleano que
# depende de $expand. Determina como se realiza la expansión (aumentando el widget de
# tamaño o no).

$caja->pack_start($boton1, $expand, $fill, $padding);
$caja->pack_start($boton2, $expand, $fill, $padding);

# Si lo queremos añadir al revés usaremos pack_end
# Finalmente, añadimos la caja a nuestra ventana principal
# Es necesario llamar al método show($widget) o show_all para que se
# nos muestren los widgets.
$ventana->add($caja);
$ventana->show_all;
```

El segundo método para colocar los widgets és Table. Como su nombre indica consiste en definar una tabla, a la cual se añadiran los widgets indicando en que casilla se colocaran (se deberá especificar lado_izquierdo, lado_derecho, arriba, abajo).

```
0          1          2
0+----------+----------+
 |          |          |
1+----------+----------+
 |          |          |
2+----------+----------+
```

```
# Creamos una tabla 2x2 sin homegeneidad ( Gtk2::Table->new($filas, $columnas,
# $homogeneidad=FALSE); )
$tabla = Gtk2::Table->new(2, 2, FALSE);

# Añadimos un widget en la columna 2, fila 1
$tabla->attach_defaults($widget, 1 , 2, 0, 1);

# Finalmente, añadimos la tabla a la ventana y mostramos todos los widgets
$ventana->add($tabla);
$ventana->show_all;
```

4.- Ventanas

Para crear una ventana simplemente tenemos que llamar al constructor *new*.

```
 my $ventana = Gtk2::Window->new;
```

El widget Window és un widget contenedor. Para añadir widgets a la ventana creada podemos usar el método *'add'*.

```
$ventana->add($listbox);
```

Veamos algunos métodos para cambiar las propiedades de las ventanas:

Establecer el título: *$ventana->set_title('Titulo de la ventana');*
Siempre encima de otras ventanas: *$ventana->set_keep_above(TRUE);*
Siempre por debajo de otras ventanas: *$ventana->set_keep_below(TRUE);*
Maximizar la ventana: *$ventana->maximize;*
Mover la ventana: *$ventana->move($xy, $y);*
Obtener su posición: *($x, $y) = $ventana->get_position;*
Establecer si se puede redimensionar: *$ventana->set_resizable(TRUE);*
Establecer si puede obtener el foco: *$ventana->set('accept-focus' => TRUE);*
Establecer el tamaño original de la ventana: *$ventana->set_default_size(640, 480);*

5.- Principales widgets

5.1.- Button

Para crear un botón podemos llamar simplemente al constructor *new* o bien a *new_with_label*;

```
$boton = Gtk2::Button->new_with_label('Boton1');
```

Los principales signals que genera el botón son: 'pressed' (cuando esta pulsado), 'clicked' (cuando se pulsa y se suelta), 'released' (cuando se deja de pulsar el botón), 'enter' (cuando el cursor entra en el área del botón) y 'leave' (cuando el cursor abandona el área del botón).

Para crear botones con imágenes y texto será necesario crear una caja horizontal (Gtk2::HBox) que contenga el Label i la imagen. Para cargar la imagen usaremos Gtk2::Image. Posteriormente, añadiremos esta caja al botón. Recordemos, que el botón también es un widget contenedor.

```
#!/usr/bin/perl -w

use Gtk2 '-init';

$ventana = Gtk2::Window->new('toplevel');
$ventana->set_border_width(20);
$hbox = Gtk2::HBox->new(0, 0);
$hbox->set_border_width(5);
$imagen = Gtk2::Image->new_from_file("carpeta.xpm");
$etiqueta = Gtk2::Label->new("Abrir");
$hbox->pack_start($imagen, FALSE, FALSE, 2);
$hbox->pack_start($etiqueta, FALSE, FALSE, 2);
$hbox->show;
$boton = Gtk2::Button->new();
$boton->add($hbox);
$ventana->add($boton);
$ventana->show_all;
Gtk2->main;
```

Botones estándard (Ok, Cancelar, Ayuda):
En vez de crear botones nuevos, podemos acudir a botones predefinidos, a través del método *'new_from_stock'*;

```
# Creamos los típicos botones Ok, Cancelar y Ayuda
$boton_ok = Gtk2::Button->new_from_stock('gtk-ok');
$boton_cancelar = Gtk2::Button->new_from_stock('gtk-cancel');
$boton_ayuda = Gtk2::Button->new_from_stock('gtk-help');
```

Toggle Buttons:
Los toggle buttons són botones con dos estados (activolinactivo).Con un click se alternan estos dos estados. En lo demás, son prácticamente iguales al botón normal.

```
$togglebutton = Gtk2::ToggleButton->new_with_label($etiqueta);

# El método get_active nos devuelve TRUE o FALSE si el botón esta activo o no.
if ($togglebutton->get_active()) {
        ....
```

```
        } else {
        ...
}

# Con set_active podemos cambiar el estado del togglebutton
$togglebutton->set_active();
```

5.2.- Frames

Los frames son un widget contenedor. Opcionalmente pueden tener un título. Son una especie de marco donde poner otros widgets.

```
# Creamos un frame con título
$frame = Gtk2::Frame->new('Titulo');
$frame->add($boton);

# Podemos alinear el título donde queramos
$frame->set_label_align ($x, $y);

# En cualquier momento podemos cambiar la etiqueta del frame
$frame->set_label($etiqueta);

# Para obtener la etiqueta del frame
$texto = $frame->get_label;
```

5.3.- Labels

Los labels son simplemente etiquetas de texto. Se utilizan para proporcionar información al usuario.

```
# Creamos un label
$etiqueta = Gtk2::Label->new('Texto del label... ');

# En cualquier momento podemos cambiar el texto del Label
$etiqueta->set_text($texto);

# Podemos obtener el texto actual del Label con:
$texto = $etiqueta->get_text;

# El label puede justificar el texto ('left', 'right', 'center');
$etiqueta->set_justify('left');
```

5.4.- Tooltips

Los tooltips son las descripciones que aparecen al situar el cursor sobre un widget. Indican para que sirve un widget.

```
# Creamos un tooltip. El objeto que nos devuleve el constructor 'new' puede
# ser usado para crear multiples tooltips.
$tooltip = Gtk2::Tooltips->new;

# Añadimos un tooltip a un botón
$tooltip->set_tip($boton, "Pulsa este botón para salir.");
```

5.5.- CheckButton

La funcionalidad del CheckButton es, prácticamente, la misma que el ToggleButton. Solo se diferencian en su aspecto exterior.

```
# Creamos un checkbutton con texto al lado
$check = Gtk2::CheckButton->new_with_label('Avisar');

# Podemos activarlo con set_active
$check->set_active(1);

# Y comprovar si esta marcado con get_active
$boolean = $check->get_active;
```

5.6.- Radiobutton

Los radiobuttons nos permiten mostrar al usuario una serie de opciones de las cuales solo podrá escoger una.
Para los radiobuttons es necesario especificar el grupo al que pertenecen (de todos los radiobuttons del mismo grupo, solamente uno podrá ser seleccionado).
Los principales signals que generan son: clicked y toggled.

Ejemplo:

```
$radiobutton1 = Gtk2::RadioButton->new_with_label('grupo', 'Madrid');
# Cogemos el grupo al que pertenece el radiobutton que acabamos de crear
@grupo = $radiobutton1->get_group;
# Creamos dos radiobuttons mas en el mismo grupo.
$radiobutton2 = Gtk2::RadioButton->new_with_label(@grupo, 'Barcelona');
$radiobutton3 = Gtk2::RadioButton->new_with_label(@grupo, 'Valencia');
```

Principales métodos del RadioButton:

- Gtk2::RadioButton->get_group;
- Gtk2::RadioButton->set_group($grupo);

5.7.- TextEntry

El widget TextEntry permite la entrada de texto por parte del usuario y la visualización del mismo en una sola linea.

El widget Entry puede llegar a producir bastantes signals. Los mas destacados son: 'move-cursor', 'activate', 'copy-clipboard','insert-at-cursor', etc.

```
# Podemos crear el Entry con 'new' o si queremos limitar el número de carácteres que el
# usuario puede entrar con 'new_with_max_length(entero)'
$entrada = Gtk2::Entry->new;
$entrada = Gtk2::Entry->new_with_max_length(40);

# Podemos añadir texto al Entry con:
$entrada->append_text($texto);

# Podemos obtener el texto que el usuario haya entrado con:
$texto = $entrada->get_text;

# Y cambiar todo el texto del Entry con:
$entrada->set_text($texto);
```

5.8.- TextView

Para lidiar con regiones de texto (+ 1 linea) tendremos que jugar, como mínimo, con dos widgets: Gtk2::TextView y Gtk2::TextBuffer. Hay muchos mas (Gtk2::TextIter, Gtk2::TextTag, Gtk2::TextMark ...), pero para visualizar poco texto, es suficiente con estos dos. Con el widget Gtk2::TextBuffer administraremos el texto.

```
# Creamos un buffer de texto
$buffer = Gtk2::TextBuffer->new;
# Insertamos el texto de la variable texto en la posición del cursor
$buffer->insert_at_cursor($texto);
# Creamos el widget visor de texto y lo llenamos con el contenido de $buffer
$visor = new Gtk2::TextView();
$visor->set_buffer($buffer)
```

6.- Menus

En Gtk2-Perl se pueden crear menus de muchas maneras. Para la creación de menus de forma fácil y rápida, recomiendo utilitzar el módulo Gtk2::SimpleMenu. Simplemente, consiste en crear una estructura con los datos que queremos que tenga el menu. Después, se crea una nueva instancia del objeto Gtk2::SimpleMenu, y le decimos que para hacer el menu utilice la estructura de datos que hemos creado.

```
$menu_arbol = [ _Fichero => {
                        item_type => '',
                        children => [
                                _Nuevo => {
                                callback => \&nuevo,
                                accelerator => '<ctrl>N',
                                },
                                _Guardar => {
                                callback => \&guardar,
                                accelerator => '<ctrl>G',
                                },
                                'G_uardar como' => {
                                callback =>\&guardarb,
                                accelerator => '<ctrl>U',
                                },
                                _Salir => {
                                callback => sub { Gtk2->main_quit; },
                                accelerator => '<ctrl>S',
                                },
                        ],
        },
        _Ayuda => {
                item_type => '',
                children => [
                _Información => {
                        callback => \&imprime,
                        accelerator => '<ctrl>I',
                        },
                A_cerca => {
                        callback => \&sobreesto,
                        accelerator => '<ctrl>O',
                        },
                ],
        },
];

# Ahora creamos un menu y le decimos que utilice esta estructura ($menu_arbol)
$menu = Gtk2::SimpleMenu->new( menu_tree => $menu_arbol );

# Y ya esta. Simplemente nos queda posicionar el menu
# que suponiendo que useis el método Table, es tan fácil como:
$tabla->attach_defaults($menu->{widget}, 0, 12, 0, 1);
```

Bueno, y aquí acabo :-). Quedan un montón de widgets y muchísimas otras cosas por ver (scrollbars, canvas, Calendarios, Diálogos, Selectores de ficheros y fuentes, bindings, Pixbuf...), pero este artículo, solo era una introducción :-).

7.- Referencias

- Información específica de Gtk2-Perl en Internet no hay mucha. Se esta empezando un tutorial, pero, de momento, esta muy incompleto (versión alpha). Lo podeis encontrar en: http://gtk2-perl.sourceforge.net/doc/gtk2-perl-tut/[7]
- Puede resultar mas útil acudir a la documentación de Gtk para C, pues es muy parecido. En algunos casos, solamente cambia el nombre de la función. Podeis encontrar el tutorial aquí: http://gtk.org/tutorial/[8]
- Otro tutorial, esta vez para el módulo Gtk1 de Perl (es bastante parecido a Gtk2-Perl), lo podeis encontrar aquí: http://personal.riverusers.com/~swilhelm/_Ë_Ferl-tutorial/[9]

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=2082&nIdPage=2
2. http://bulma.net/body.phtml?nIdNoticia=2082&nIdPage=3
3. http://bulma.net/body.phtml?nIdNoticia=2082&nIdPage=4
4. http://bulma.net/body.phtml?nIdNoticia=2082&nIdPage=5
5. http://bulma.net/body.phtml?nIdNoticia=2082&nIdPage=6
6. http://www.cpan.org
7. http://gtk2-perl.sourceforge.net/doc/gtk2-perl-tut/

8. http://gtk.org/tutorial/
9. http://personal.riverusers.com/~swilhelm/gtkperl-tutorial/

E-mail del autor: vmiralda _ARROBA_ parcfluvial.org