El libro de EWL

dan 'dj2' sinclair

El libro de EWL

by dan 'dj2' sinclair

Este libro es un tutorial sobre el uso de EWL (Enlightened Widget Library).

Table of Contents

1. Introducción 1
2. Empezando 2
    - Instalando EWL 2
    - Creando una ventana simple 2
    - Hola mundo 4
    - Callbacks 8
3. Jerarquía de objectos 10
    - Introducción 10
    - Casteado de objectos 11
    - Añadiendo nuevos widgets 11
4. Empaquetado de widgets 12
5. Configuración 13
    - Configuración EWL 13
    - Configuración de la aplicación 13
6. Temas EWL 15
7. Widgets 16
    - ewl_hbox and ewl_vbox 16
    - ewl_button 16
    - ewl_checkbutton 17
    - ewl_combo 18
    - ewl_dialog 19
    - ewl_entry 21
    - ewl_filedialog 22
    - ewl_image 23
    - ewl_menu 24
    - ewl_notebook 24
    - ewl_password 24
    - ewl_progressbar 25
    - ewl_radiobutton 25
    - ewl_scrollpane 26
    - ewl_seeker 26
    - ewl_spinner 26
    - ewl_table 27
    - ewl_text 27
    - ewl_tooltip 28
    - ewl_tree 28
    - ewl_media 28
    - ewl_window 30
8. Contribuyendo 33
A. Ejemplo de reproductor de media EWL 34

List of Figures

3.1. La jerarquía de objetos EWL 10
4.1. El relleno y los Insets 12
7.1. Un botón EWL 17
7.2. Un botón de opción EWL 17
7.3. Una combobox EWL 18
7.4. Un diálogo EWL 19
7.5. Una caja de entrada EWL 21
7.6. Un diálogo de archivo EWL 22
7.7. Un libro de notas EWL 24
7.8. Un diálogo de contraseña EWL 25
7.9. Una barra de progreso EWL 25
7.10. Un radiobutton EWL 25
7.11. Un buscador EWL 26
7.12. Un spinner EWL 26
7.13. Un tooltip EWL 28
7.14. Un objeto media EWL 28
7.15. Una ventana EWL 30

List of Examples

3.1. Creando widgets EWL 11
7.1. Creando cajas EWL 16
7.2. Creating a button 17
7.3. Callback de botón 17
7.4. Creando un checkbutton 17
7.5. Callback del botón 18
7.6. Creating a combo box 18
7.7. callback de cambio de valor combobox 19
7.8. código de diálogo EWL 19
7.9. Callback de diálogo EWL 20
7.10. Creando una caja de entrada EWL 21
7.11. callback de cambio de valor para Ewl_Entry 21
7.12. Creando un diálogo de archivo EWL 23
7.13. callback para Ewl_Filedialog open 23
7.14. Ewl_Image 23
7.15. Creando un diálogo de contraseña EWL 25
7.16. callback de cambio de valor de Ewl_Password 25
7.17. Creando un buscador EWL 26
7.18. Ewl_Seeker callback 26
7.19. Código Ewl_Text 27
7.20. Ewl_Media code 29
7.21. Ewl_Media callbacks 29
7.22. Creando una ventana EWL 31
7.23. Ewl Window destroy callback 31
A.1. Reproductor de media EWL 34

Chapter 1. Introducción

La EWL (Enlightened Widget Library) es una librería de creación de interfaces de usuario basada en las EFL (Enlightenment Foundation Libraries).

El autor principal de EWL es:

- Nathan 'RbdPngn' Ingersoll

EWL funciona de manera similar a otras librerías de widgets, dado que está basada en un sistema de callback. Conforme los elementos son creados y añadidos al interfaz, los callbacks deseados son registrados, estas funciones serán activadas cuando el evento especificado ocurra.

Este tutorial es un intento de familiarizar al usuario con los diferentes aspectos del sistema EWL. El tutorial probablemente nunca documentará todos los aspectos de EWL conforme el sistema continue creciendo. Una buena comprensión de la programación en C es asumida durante el tutorial.

Si tienes cualquier problema con este tutorial, o el uso de EWL en general, cualquier feedback es grandemente appreciado puesto que ayudará a mejorar el tutorial o la propia EWL. Por favor mira la sección Contribuyendo para más información.

Chapter 2. Empezando

Instalando EWL

Antes de usar EWL necesitas tener las librerías instaladas en tu equipo. EWL puede ser obtenido del CVS en Enlightenment y instrucciones de como esto se hace se pueden encontrar en: `http://www.enlightenment.org/pages/source.html [http://www.enlightenment.org/pages/source.html]` junto con detalladas instrucciones de instalación.

Necesitarás instalar un montón de dependencias antes de ser capaz de instalar EWL, esto es porque depende de que tantas de las EFL librerías EFL estén presentes en el sistema.

Una vez que tengas las otras librer#as EFL instaladas, instalar EWL es tan simple como:

```
./configure;
make;
sudo make install;
```

Creando una ventana simple

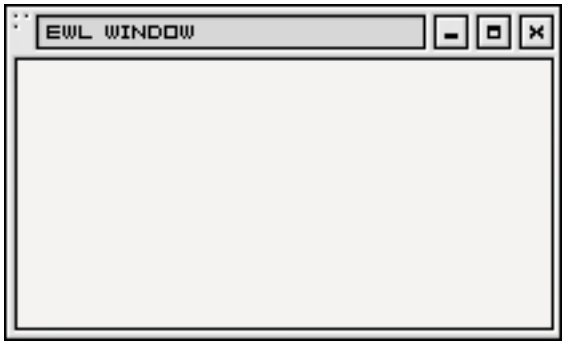
El primer paso en crear una aplicación EWL es obtener la ventana principal para mostrarla en la pantalla.

```
#include <Ewl.h>

void destroy_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_destroy(w);
    ewl_main_quit();
}

int main(int argc, char ** argv) {
    Ewl_Widget *win = NULL;

    if (!ewl_init(&argc, argv)) {
        printf("Unable to init ewl\n");
        return 1;
    }

    win = ewl_window_new();
    ewl_windowt_title_set(EWL_WINDOW(win), "EWL Window");
    ewl_window_name_set(EWL_WINDOW(win), "EWL_WINDOW");
    ewl_window_class_set(EWL_WINDOW(win), "EWLWindow");
    ewl_object_size_request(EWL_OBJECT(win), 200, 100);
    ewl_callback_append(win, EWL_CALLBACK_DELETE_WINDOW, destroy_cb, NULL);
    ewl_widget_show(win);

    ewl_main();
    return 0;
}
```

Este programa puede ser compilado con un simple:

```
zero@oberon [create_window] -> gcc -o create_window main.c \
 `ewl-config --cflags --libs`
```

Y cuando ise ejecuta debería producir algo similar a:



Ahora que sabemos lo que estamos haciendo, veamos el código con un poco mas de detalle.

```
#include <Ewl.h>
```

Todas las aplicaciones EWL empezarán con el include <Ewl.h>. Esto introducirá todos los otros archivos de cabecera que EWL requiera para funcionar.

```
void destroy_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_destroy(w);
    ewl_main_quit();
}
```

`destroy_cb` ser# usada por EWL cuando el gestor de ventanas requiera que la aplicaci#n termine. Los callbacks serán descritos mas tarde en la sección Callbacks section.

`ewl_widget_destroy()` es usado para señalar a EWL que ya no necesitamos el objeto dado, en este caso la ventana, y para que EWL limpie los recurson usados por ese widget.

Finalmente, llamamos a `ewl_main_quit()` que causa que EWL salga de su ciclo principal de proceso y vuelva de la función `ewl_main()`.

```
int main(int argc, char ** argv) {
    Ewl_Widget *win = NULL;

    if (!ewl_init(&argc, argv)) {
        printf("Unable to init ewl\n");
        return 1;
    }
```

Antes de que podamos usar EWL debemos iniciar la librería.Esto se hace mediante la llamada a `ewl_init()`. Pasamos los parámetros argc y argv desde main a EWL dado que hay unas cuantas opciones específicas para EWL entre los argumentos.

Estos argumentos incluyen:

Opciones de linea de comandos EWL

- --ewl-theme <name>
- --ewl-segv
- --ewl-software-x11
- --ewl-gl-x11
- --ewl-fb

El parámetro <name> para la opción --ewl-theme es el nombre del tema que deseas usar. Este puede estar localizado en uno de los directorios del sistema, o en el directorio local.

Si EWL fué capaz de inicializarse con éxito la llamada a `ewl_init()` devolverá un valor > 0. Si no hubo éxito no tiene realmente sentido continuar dado que EWL no funcionará correctamente.

```
    win = ewl_window_new();
    ewl_window_title_set(EWL_WINDOW(win), "EWL Window");
    ewl_window_name_set(EWL_WINDOW(win), "EWL_WINDOW");
    ewl_window_class_set(EWL_WINDOW(win), "EWLWindow");
    ewl_object_size_request(EWL_OBJECT(win), 200, 100);
    ewl_callback_append(win, EWL_CALLBACK_DELETE_WINDOW, destroy_cb, NULL);
    ewl_widget_show(win);
```

Aquí es donde la ventana es creada. Una llamada a `ewl_window_new()` crea la nueva ventana vacia. Entonces tomamos esa ventana y empezamos a añadir datos. Empezamos por colocar el título con `ewl_window_title_set()`, que colocará la cadena que el gestor de ventanas mostrará en la parte superior de la ventana. Las siguientes dos llamadas, `ewl_window_name_set()` y `ewl_window_class_set()`, inician datos que serán usados por el gestor de ventanas para gestionar mejor tu aplicación.

Procedemos entonces a iniciar el tamaño base de la ventana con una llamada a `ewl_object_size_request()`. Los parámetros segundo y tercero (200 , 100) especifican la anchura y altura con las que queremos crear la ventana. Notarás que esta llamada es casteada a `EWL_OBJECT()`. Esto es a causa de la jerarquía de widgets que EWL provee (descrita más a fondo en el capítulo Object Hierarchy). Un ewl_window es un ewl_object así que podemos usar las operaciones de ewl_object en una ewl_window.

Entonces procedemos a añadir el callback delete a la ventana mediante una llamada a `ewl_callback_append`. El segundo parámetro de la cual es el tipo de señal que queremos recibir, el tercero es la función a la que llamar y finalmente el cuarto son cualesquiera datos de usuario que hayan de ser enviados al callback.

Una vez la ventana está iniciada y lista para empezar, una simple llamada a `ewl_widget_show()` hará que EWL muestre la ventana.

```
    ewl_main();
    return 0;
}
```

La llamada a `ewl_main()` le dir# a EWL que empiece su ciclo de proceso principal esperando por señales. `ewl_main()` se encargará del apagado de EWL cuando se salga del ciclo de proceso principal.

Eso es todo. Aunque es probablemente una de las aplicaciones EWL más simples que pueden producirse, será usada como la base para muchos de los otros ejemplos presentados en este tutorial, y muchas aplicaciones EWL que son producidas.

Hola mundo

Cuando tienes una ventana en la pantalla es hora de hacer algo más divertido con ella. Siguiendo la gran tradición , algo con Hola.

Solo voy a explicar porciones del programa que no hayan sido vistas ya. Si hay algo que no entiendes por favor busca en la sección anterior y debería estar explicado allí I am only going to explain the portions of the program which have not already been seen. If there is something you do not understand please reference the previous section and it should be explained there.

```
#include <stdio.h>
#include <Ewl.h>

void destroy_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_destroy(w);
    ewl_main_quit();
}

void text_update_cb(Ewl_Widget *w, void *event, void *data) {
    char *s = NULL;
    Ewl_Widget *label = NULL;
    char buf[BUFSIZ];

    s = ewl_entry_get_text(EWL_ENTRY(w));
    label = (Ewl_Widget *)data;

    snprintf(buf, BUFSIZ, "Hello %s", s);
    ewl_text_text_set(EWL_TEXT(label), buf);

    free(s);
    return;
}

int main(int argc, char ** argv) {
    Ewl_Widget *win = NULL;
    Ewl_Widget *box = NULL;
    Ewl_Widget *label = NULL;
    Ewl_Widget *o = NULL;

    /* init the library */
    if (!ewl_init(&argc, argv)) {
        printf("Unable to initialize EWL\n");
        return 1;
    }

    /* create the window */
    win = ewl_window_new();
    ewl_window_title_set(EWL_WINDOW(win), "Hello world");
    ewl_window_class_set(EWL_WINDOW(win), "hello");
    ewl_window_name_set(EWL_WINDOW(win), "hello");
    ewl_object_size_request(EWL_OBJECT(win), 200, 50);
    ewl_callback_append(win, EWL_CALLBACK_DELETE_WINDOW, destroy_cb, NULL);
    ewl_widget_show(win);

    /* create the container */
    box = ewl_vbox_new();
    ewl_container_child_append(EWL_CONTAINER(win), box);
    ewl_object_fill_policy_set(EWL_OBJECT(box), EWL_FLAG_FILL_ALL);
    ewl_widget_show(box);

    /* create text label */
    label = ewl_text_new(NULL);
    ewl_container_child_append(EWL_CONTAINER(box), label);
    ewl_object_alignment_set(EWL_OBJECT(label), EWL_FLAG_ALIGN_CENTER);
```

```
    ewl_text_style_set(EWL_TEXT(label), "soft_shadow");
    ewl_text_color_set(EWL_TEXT(label), 255, 0, 0, 255);
    ewl_text_text_set(EWL_TEXT(label), "hello");
    ewl_widget_show(label);

    /* create the entry */
    o = ewl_entry_new("");
    ewl_container_child_append(EWL_CONTAINER(box), o);
    ewl_object_alignment_set(EWL_OBJECT(o), EWL_FLAG_ALIGN_CENTER);
    ewl_object_padding_set(EWL_OBJECT(o), 5, 5, 5, 0);
    ewl_text_color_set(EWL_TEXT(EWL_ENTRY(o)->text), 0, 0, 0, 255);
    ewl_callback_append(o, EWL_CALLBACK_VALUE_CHANGED, text_update_cb, label);
    ewl_widget_show(o);

    ewl_main();
    return 0;
}
```

Si compilas y corres esta aplicación de la misma manera que el primer ejemplo deberías ver algo similar a la siguiente ventana.

Esto es un poco mas largo que la simple creación de una ventana, pero tambien incluye mas funcionalidad. Si corres este programa deberías ver una ventana simple con un poco de texto diciendo "Hello" en la parte superior y un área de texto. Teclear en el área de texto y pulsar "Enter" mostrará "Hello" más lo que hayas tecleado.

A la cadena "Hola" se le ha aplicado un poco de estilización. Ouedes ver que al texto se le ha aplicado un simple cambio de color y es mostrado con una sombra.

Ahora que sabes lo que hace, vamos a hechar un vistazo a los nuevos pedazos de código que introduce este ejemplo.

```
void text_update_cb(Ewl_Widget *w, void *event, void *data) {
    char *s = NULL;
    Ewl_Widget *label = NULL;
    char buf[BUFSIZ];

    s = ewl_entry_get_text(EWL_ENTRY(w));
    label = (Ewl_Widget *)data;

    snprintf(buf, BUFSIZ, "Hello %s", s);
    ewl_text_text_set(EWL_TEXT(label), buf);

    free(s);
    return;
}
```

`text_update_cb()` es el callback que vamos a registrar para cuando el usuario ha presionado "Enter" en el campo de texto. Tiene la misma firma que el callback de destroy, y todos los otros callbacks de EWL que estaremos registrando.

El texto que ha sido introducido es recuperado con una llamada a`ewl_entry_get_text()` pasando el widget de entrada desde el cual queremos recuperar el texto. Esto retornarña un puntero a la cadena de texto, es responsabilidad de la aplicación liberar este puntero.

Entonces casteamos el parámetro `data` a `Ewl_Widget`. Esto es porque, como verás en el callback register, estamos añadiendo un widget a este callback como parámetro de datos.

Entonces podemos tomar este nuevo texto y reemplazar los contenidos de la etiqueta de texto actual llamando a `ewl_text_text_set()` pasándole el objeto de texto y el texto a ser mostrado.

```
box = ewl_vbox_new();
ewl_container_append_child(EWL_CONTAINER(win), box);
ewl_object_fill_policy_set(EWL_OBJECT(box), EWL_FLAG_FILL_ALL);
ewl_widget_show(box);
```

Aunque podríamos añadir cualquier widget a la ventana principal de la aplicación, es un poco mas limpio añadirlos a una caja que se añade a la ventana principal. La caja es creada con una llamada a `ewl_vbox_new()` creando una disposición de caja vertical. Podríamos haber usado `ewl_hbox_new()` si quisieramos una caja horizontal en vez de una vertical. En cuanto la caja es creada la añadimos a la ventana llamando a `ewl_container_child_append()`. Esto coloca el widget dado como siguiente elemento en el contenedor. Los contenedores son empaquetados desde arriba hacia abajo, o de izquierda a derecha, así que el orden en que los elementos son insertados en el contenedor afecta su apariencia en la pantalla. Finalmente, antes de enseñar el widget, seleccionamos una política de relleno con `ewl_object_fill_policy_set()`. La política de relleno cambia la manera en que el objeto rellena su espacio disponible.

Las políticas de relleno posibles son:

EWL Fill Flags

- EWL_FLAG_FILL_NONE
- EWL_FLAG_FILL_HSHRINK
- EWL_FLAG_FILL_VSHRINK
- EWL_FLAG_FILL_SHRINK
- EWL_FLAG_FILL_HFILL
- EWL_FLAG_FILL_VFILL
- EWL_FLAG_FILL_FILL
- EWL_FLAG_FILL_ALL

Todas las cuales deberían tener un significado obvio, con las excepciones de `EWL_FLAG_FILL_SHRINK`, `EWL_FLAG_FILL_FILL` y `EWL_FLAG_FILL_ALL`. La opción SHRINK es el or lógico de las dos opciones HSHRINK y VSHRINK . La opción FILL es el or lógico de las dos opciones HFILL y VFILL. Finalmente la opción ALL es el or lógico de las dos opciones SHRINK y FILL.

```
label = ewl_text_new(NULL);
ewl_container_child_append(EWL_CONTAINER(box), label);
ewl_object_alignment_set(EWL_OBJECT(label), EWL_FLAG_ALIGN_CENTER);
ewl_text_style_set(EWL_TEXT(label), "soft_shadow");
ewl_text_color_set(EWL_TEXT(label), 255, 0, 0, 255);
ewl_text_text_set(EWL_TEXT(label), "Hello");
ewl_widget_show(label);
```

Ahora que tenemos nusetra caja contenedor iniciada, creamosel elemento de texto que va a funcionar

como nuestra etiqueta. La etiqueta es creada con una llamada a `ewl_text_new()`. En este caso, pasamos NULL como valor porque especificaremos nuestro texto después de añadir algún estilizado al objeto. También puedes pasar una cadena de texto a `ewl_text_new()` si es necesario. Recuerda que el estilizado de texto ocurre para el texto que es añadido *después* de que se añada el estilizado.

Cuando el widget es creado lo añadimos a la caja con `ewl_container_child_append()`. Después seleccionamos la alineación del objeto de texto por medio de `ewl_object_alignment_set()`. Esto especifica como se alinearán los contenidos dentro del propio widget.

La función de alineación aceptará uno de:

Opciones de alineación EWL

- EWL_FLAG_FILL_CENTER
- EWL_FLAG_FILL_LEFT
- EWL_FLAG_FILL_RIGHT
- EWL_FLAG_FILL_TOP
- EWL_FLAG_FILL_BOTTOM

Una vez todas las opciones de widget han sido especificadas, añadimos algunas propiedades de formato de texto al widget. La primera, `ewl_text_style_set()`, coloca el estilo del objeto de texto. Los estilos cambian la apariencia del texto aplicando algún tipo de filtro, en este caso, creando una apariencia de "sombra suave" para el widget. Seleccionamos entonces el color del texto rojo llamando a `ewl_text_color_set()`. Hay cuatro parámetros para la función de color, siendo estos rojo, verde, azul, y alpha.

```
o = ewl_entry_new("");
ewl_container_append_child(EWL_CONTAINER(box), o);
ewl_object_alignment_set(EWL_OBJECT(o), EWL_FLAG_ALIGN_CENTER);
ewl_object_padding_set(EWL_OBJECT(o), 5, 5, 5, 0);
ewl_text_color_set(EWL_TEXT(EWL_ENTRY(o)->text), 0, 0, 0, 255);
ewl_callback_append(o, EWL_CALLBACK_VALUE_CHANGED, text_update_cb, label);
ewl_widget_show(o);
```

El widget final que crearemos es una caja de entrada de texto. Esto es hecho con una llamada a `ewl_entry_new()`. En este caso estamos dando "" como valor, pero podíamos dar una cadena inicial para mostrar en la entrada de texto. Hacemos un conjunto semejante de inicializaciones a la caja de entrada, seleccionando la alineación y seleccionando el color del texto negro. La llamada a `ewl_object_padding_set()` selecciona la cantidad de padding alrededor del widget. Los cuatro parámetros son izquierda, derecha, arriba y abajo.

Con eso deberías tener una comprensión básica de las funciones EWL y como varios widgets son creados y configurados.

Callbacks

La EWL funciona mediante el uso de callbacks. Una gran cantidad del trabajo interno de la propia librería también trabaja con callbacks.

Un callback es una función que será llamada cuando ocurra un evento específico. Estos eventos pueden ser cualquier cosa desde el usuario pulsando un botón hasta la ventana siendo destruída por el gestor de ventanas.

Para los eventos de los cuales la aplicación requiere una notificación se registra un callback a través de la EWL. Esto es hecho con `ewl_callback_append()`. Esta función toma cuatro parámetros: el objeto al que añadir el callback, el callback deseado, la función de callback, y los datos de usuario.

Algunos de los callbacks posibles incluyen:

Callbacks EWL posibles

EWL_CALLBACK_DESTROY	El widget es liberado
EWL_CALLBACK_DELETE_WINDOW	La ventana está siendo cerrada.
EWL_CALLBACK_KEY_DOWN	Una tecla fué presionada.
EWL_CALLBACK_KEY_UP	Una tecla fué soltada.
EWL_CALLBACK_MOUSE_DOWN	El botón del ratón fué presionado.
EWL_CALLBACK_MOUSE_UP	El botón del ratón fue soltado.
EWL_CALLBACK_MOUSE_MOVE	El ratón fué movido.
EWL_CALLBACK_MOUSE_WHEEL	La rueda del ratón se desplazó
EWL_CALLBACK_FOCUS_IN	El ratón fué posicionado sobre el widget.
EWL_CALLBACK_FOCUS_OUT	El ratón se movió fuera del widget.
EWL_CALLBACK_SELECT	El widget fué seleccionado mediante el ratón o el teclado.
EWL_CALLBACK_DESELECT	El widget fué deselecionado por el ratón o el teclado.
EWL_CALLBACK_CLICKED	El ratón fué presionado y soltado en un widget.
EWL_CALLBACK_DOUBLE_CLICKED	El ratón fué clickado dos veces rápidamente.
EWL_CALLBACK_HILITED	El ratón está sobre el widget.
EWL_CALLBACK_VALUE_CHANGED	El valor en el widget cambió.

La función callback tiene una firma como `void fcn(Ewl_Widget *, void *, void *)`. El primer parámetro es el widget que activó este callback. El segundo parámetro son los datos del evento y el tercer parámetro son los datos de usuario añadidos.

Los datos de evento son un tipo que se refiere al propio callback. Así, por ejemplo, cuando el callback para `EWL_CALLBACK_MOUSE_WHEEL` es llamado los datos de evento tendrán una estructura de tipo `Ewl_Event_Mouse_Wheel` y esta estructura contiene información adicional sobre el evento. En el caso de la rueda, los modificadores de teclado, la posición del ratón y la dirección del desplazamiento.

El último parámetro para la función son los datos de usuario. Esto te permite añadir cualesquiera dsatos que quieras pasar al callback cuando sea ejecutado. Estos datos serán provistos al callback como su tercer parámetro.

Chapter 3. Jerarquía de objectos

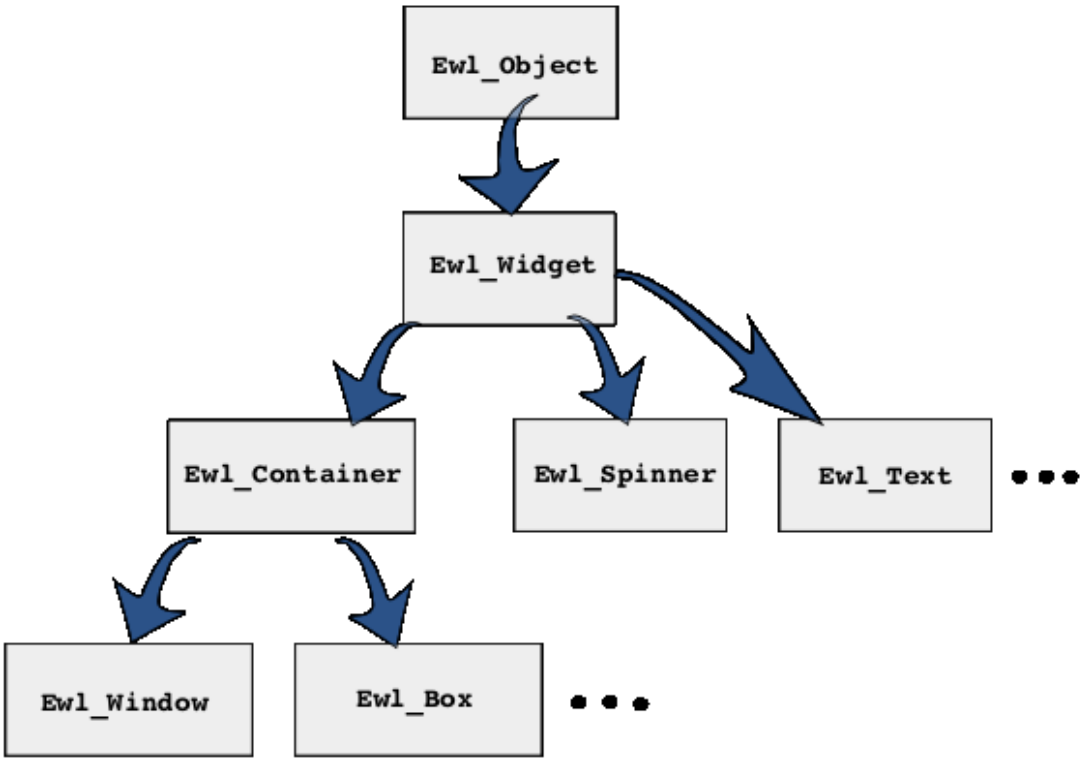
Introducción

Los widgets EWL forman una jerarquía. El widget base del que todo deriva es el `Ewl_Object`. `Ewl_Object` provee toda la funcionalidad base para cada widget incluyendo el dimensionado, alineación, políticas de relleno, relleno y otras. Este es el principal bloque constructivo de la EWL. Una aplicación que use EWL nunca necesitará obtener un `Ewl_Object`.

Justo sobre el `Ewl_Object` está el `Ewl_Widget`. De nuevo, todos los widgets heredan de `Ewl_Object`. Este objecto provee la funcionalidad base para que un widget ineractúe con el usuario. Como con `Ewl_Object` una aplicación nunca necesitará obtener un `Ewl_Widget`.

Con el `Ewl_Widget` en su lugar podemos empezar a construir la jerarquía de objetos que forman la EWL. La jerarquía es similar a la mostrada en la figura EWL Object Hierarcy más abajo.

Figure 3.1. La jerarquía de objetos EWL



El objeto `Ewl_Container` es construido sobre el objeto `Ewl_Widget` y provee la funcionalidad para widgets que tienen que contener otros widgets. Esto incluye cualquier cosa desde la ventana principal, a cajas, a paneles de desplazamiento.

Casteado de objectos

Conforme progreses en la EWL notarás que hay un montón de casting entre los diferentes tipos. Para hacer esto más fácil, cada cast a un tipo particular tiene una macro EWL_TYPE() definida. Así por ejemplo hay definidas EWL_OBJECT(o) y EWL_WIDGET(o) para hacer la vida más fácil.

Estas macros siempre deberían ser usadas cuando se convierta entre widgets EWL para saber que se está haciendo lo correcto.

Añadiendo nuevos widgets

Para añadir widgets nuevos en EWL necesitas crear una nueva struct que tenga el tipo apropiado de subclase como primer elemento. Este objeto de subclase no debe ser un puntero.

Example 3.1. Creando widgets EWL

```
struct Ewl_Foo {
    Ewl_Container container;
    int bar;
}
```

Esto creará un nuevo widget Ewl_Foo que hereda de `Ewl_Container` así que deberías ser capaz de empaquetar otros widgets en este nuevo tipo de widget.

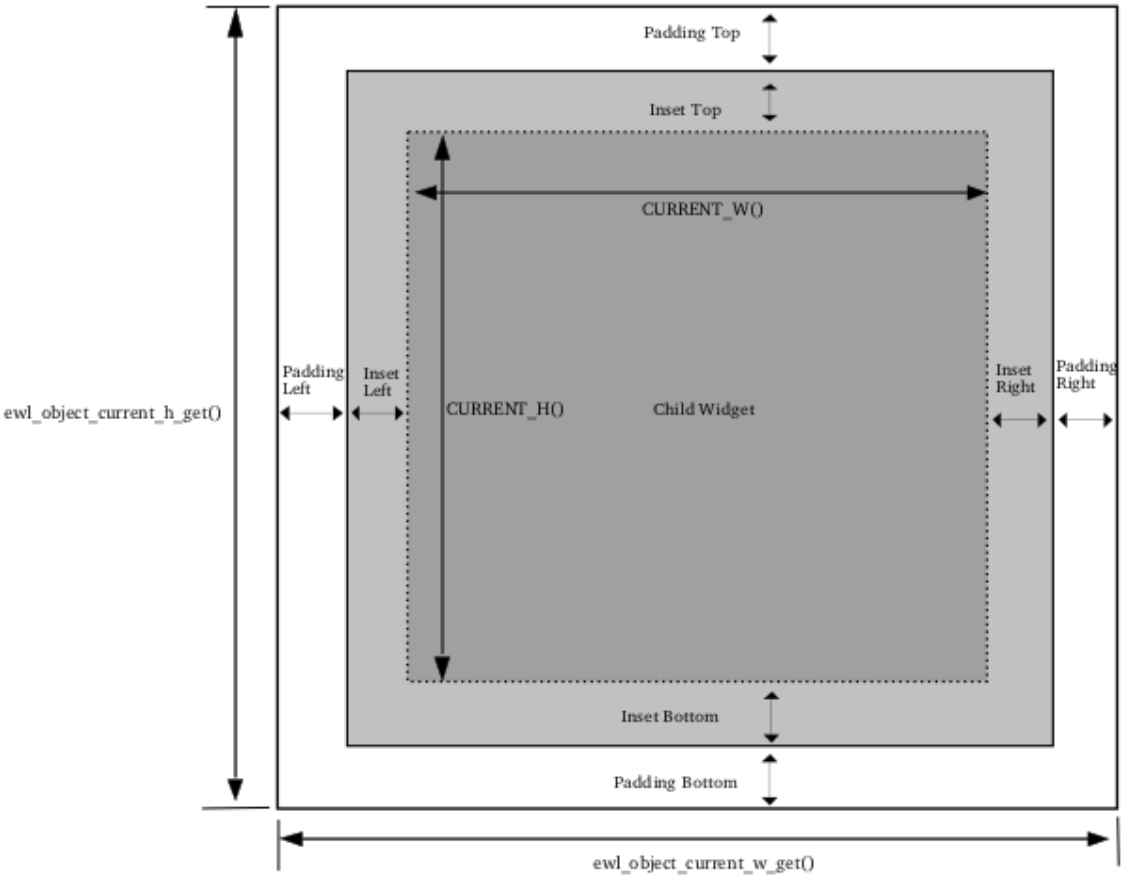
Chapter 4. Empaquetado de widgets

Conforme escribas una aplicación EWL necesitarás empezar colocando los widgets en las distintas cajas. Para hacerlo, necesitarás algo de información sobre como EWL empaqueta los widgets.

Hay dos orientaciones principales para contenedores en EWL, vertical u horizontal. Las diferentes orientaciones pueden ser vistas fácilmente en the section called “ewl_hbox and ewl_vbox”.

En EWL, cada widget tiene una cantidad de relleno alrededor del widget y un inset vinculado al widget. Esto se ve abajo en Figure 4.1, “El relleno y los Insets”.

Figure 4.1. El relleno y los Insets



Chapter 5. Configuración

Configuración EWL

EWL usa el sistema Ecore_Config para manejar todos sus datos de configuración. Esto hace fácil el cambio de valores mediante las herramientas existentes para trabajar con Ecore_Config.

Las siguientes son las claves usadas por EWL con una descripción breve.

/ewl/debug/enable	Habilita el modo de depuración
/ewl/debug/level	Selecciona el nivel de depuración [0 - 10]
/ewl/evas/render_method	Selecciona el método que Evas usará para renderizar. Este puede ser uno de: • software_x11 • gl_x11 • fb para X11 software, X11 OpenGL y Framebuffer respectivamente.
/ewl/evas/font_cache	El tamaño de la caché de fuentes de Evas
/ewl/evas/image_cache	El tamaño de la caché de imagenes de Evas
/ewl/theme/name	El nombre del tema EWL a usar (menos la porción .eet)
/ewl/theme/cache	Un booleano para indicar si los valores del tema deberían ser cacheados por EWL
/ewl/theme/color_classes/override	Sustituye las clases de color por defecto
/ewl/theme/color_classes/count	El número de clases de color que son sustituídas
/ewl/theme/color_class/[n]/name	El nombre de la clase de color número [n]
/ewl/theme/color_class/[n][rgba]	Una clave para cada uno de los valores de color r, g, b, a

Configuración de la aplicación

La mejor manera para una aplicación de manejar su configuración específica es usar también Ecore_Config. Hacerlo es simple y ya maneja cosas como valores por defecto y callbacks para cambios de datos.

Como medida de precaución probablemente deberías hacer una llamada a `ecore_init()` en tu código antes de usar las funciones Ecore_Config. Esto garantizará que Ecore no será apagado antes de que hayas terminado de usarlo (Necesitarás hacer una llamada a `ecore_shutdown()` cuando hayas terminado).

Antes de empezar a usar Ecore_Config debes hacer una llamada a `int ecore_config_init(char *)` donde el parámetro es el nombre bajo el cual quieras que tu configuración aparezca en Ecore_Config. Este es también el nombre que sería usado con **examine** para

cambiar tus datos de configuración. Cuando hayas terminado de usar Ecore_Config deberías llamar a `int ecore_config_shutdown(void)` para cerrar el sistema Ecore_Config.

Chapter 6. Temas EWL

EWL usa la librería Edje para manejar todos sus requerimientos de theming. Los temas pueden estar definidos el el archivo de configuración o en la línea de comandos usando la opción `--ewl-theme`.

El uso de Edje para theming proporciona a EWL una gran flexibilidad y provee al diseñador de temas con un gran poder para personalizar el aspecto de EWL.

Chapter 7. Widgets

Miraremos ahora a cada widget individualmente. Mira el código que crea el widget y una captura de pantalla del widget en acción (si es aplicable).

ewl_hbox and ewl_vbox

Los widgets caja te permiten especificar distintas maneras en que la aplicación será colocada. Puedes crear una caja horizontal (hbox) o vertical (vbox). Una caja horizontal tendrá a sus widgets hijos empaquetados desde la izquierda hacia la derecha mientras que una caja vertical tendrá sus widgets empaquetados de arriba abajo.

Un widget caja no se mostrará en la propia aplicación, es usado solo como un contenedor para otros widgets.

Example 7.1. Creando cajas EWL

```
Ewl_Widget *hbox = ewl_hbox_new();
ewl_widget_show(hbox);

Ewl_Widget *vbox = ewl_vbox_new();
ewl_widget_show(vbox);
```

Los widgets de caja son relativamente simples de crear y usar, sólamente requiriendo una llamada a la función new.

Las funciones para manipular cajas incluyen:

- `void ewl_box_set_orientation(Ewl_Box *, Ewl_Orientation)`
- `Ewl_Orientation ewl_box_orientation_get(Ewl_Box *)`
- `void ewl_box_spacing_set(Ewl_Box *, int)`
- `void ewl_box_homogeneous_set(Ewl_Box *, int)`

La opción `Ewl_Orientation` puede ser uno de:

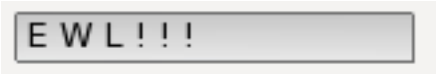
- EWL_ORIENTATION_HORIZONTAL
- EWL_ORIENTATION_VERTICAL

`ewl_box_set_spacing()` seleccionará la cantidad de espacio entre los elementos en la caja con el valor dado. `ewl_box_set_homogeneous()` configurará la caja para dar a todos los elementos en ella un tamaño igual si recibe true, sino tendrán su tamaño requerido.

ewl_button

El widget botón es simplemente un widget con una etiqueta incluída. Cuando el usuario pulsa en el botón el callback vinculado a `EWL_CALLBACK_CLICKED` será ejecutado.

Figure 7.1. Un botón EWL



Example 7.2. Creating a button

```
    Ewl_Widget *button = ewl_button_new("A button");
    ewl_object_alignment_set(EWL_OBJECT(button), EWL_FLAG_ALIGN_CENTER);
    ewl_callback_append(button, EWL_CALLBACK_CLICKED, button_cb, NULL);
    ewl_widget_show(button);
```

La porción de etiqueta del botón puede ser alineada a cualquiera de los valores `EWL_FLAG_ALIGN_*`.

Example 7.3. Callback de botón

```
void button_cb(Ewl_Widget *w, void *event, void *data) {
    printf("button pressed\n");
}
```

La etiqueta de un botón puede ser manipulada después de que el botón ha sido creado mediante las dos llamadas:

- `char *ewl_button_label_get(EwlButton *)`
- `void ewl_button_label_set(EwlButton *, char *)`

ewl_checkbutton

Figure 7.2. Un botón de opción EWL

Example 7.4. Creando un checkbutton

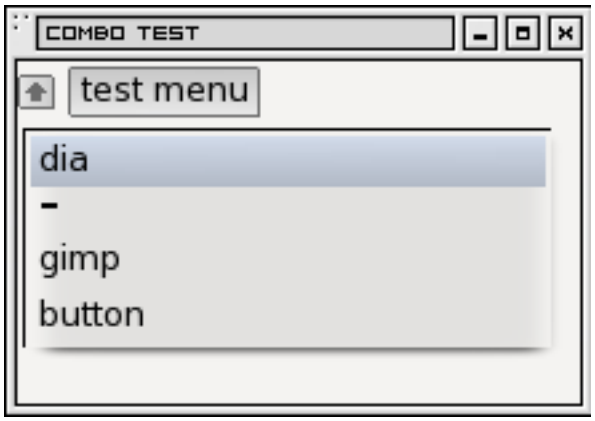
```
    Ewl_Widget *cb = ewl_checkbutton_new("Label");
    ewl_checkbutton_label_position_set(EWL_CHECKBUTTON(cb), EWL_FLAG_ALIGN_LEFT);
    ewl_callback_append(cb, EWL_CALLBACK_VALUE_CHANGED, checkbutton_cb, NULL);
    ewl_widget_show(cb);
```

Example 7.5. Callback del botón

```
void checkbutton_cb(Ewl_Widget *w, void *event, void *data) {
    if (ewl_checkbutton_is_checked(EWL_CHECKBUTTON(w)))
        printf("checked\n");
    else
        printf("Not checked\n");
}
```

ewl_combo

Figure 7.3. Una combobox EWL



Example 7.6. Creating a combo box

```
    Ewl_Widget *combo = ewl_combo_new("combo box");
    ewl_callback_append(combo, EWL_CALLBACK_VALUE_CHANGED,
                                    combo_change_cb, NULL);
    ewl_widget_show(combo);

    Ewl_widget *item1 = ewl_menu_item_new(NULL, "foo");
    ewl_container_child_append(EWL_CONTAINER(combo));
    ewl_widget_show(item1);
```

Example 7.7. callback de cambio de valor combobox

```
void combo_change_cb(Ewl_Widget *w, void *event, void *data) {
    char *text = (char *)event;
    printf("Value changed to %s\n", text);
}
```

ewl_dialog

El widget `Ewl_Dialog` provee una manera de mostrar al usuario una simple caja de diálogo que puede esperar una respuesta, dar avisos, o simplemente mostrar mensajes.

Figure 7.4. Un diálogo EWL

Example 7.8. código de diálogo EWL

```
    Ewl_Widget *dialog = NULL;
    Ewl_Widget *o = NULL;

    o = ewl_text_new("a dialog eh");
    ewl_object_alignment_set(EWL_OBJECT(o),
    EWL_FLAG_ALIGN_CENTER);
    ewl_widget_show(o);

    dialog = ewl_dialog_new(EWL_POSITION_BOTTOM);
    ewl_dialog_set_has_separator(EWL_DIALOG(dialog), 0);
    ewl_dialog_add_widget(EWL_DIALOG(dialog), o);
    ewl_object_alignment_set(EWL_OBJECT(dialog), EWL_FLAG_ALIGN_CENTER);
    ewl_widget_show(dialog);

    o = ewl_dialog_set_button(EWL_STOCK_OK, EWL_RESPONSE_OK);
    ewl_container_child_append(EWL_CONTAINER(dialog), o);
    ewl_callback_append(o, EWL_CALLBACK_CLICKED, dialog_clicked_cb, dialog);
    ewl_widget_show(o);
```

```
o = ewl_dialog_set_button(EWL_STOCK_CANCEL, EWL_RESPONSE_CANCEL);
ewl_container_child_append(EWL_CONTAINER(dialog), o);
ewl_callback_append(o, EWL_CALLBACK_CLICKED, dialog_clicked_cb, dialog);
ewl_widget_show(o);
```

Este ejemplo creará un `Ewl_Dialog` con dos botones: un botón de OK y un botón de Cancel. El própio diálogo es creado con la llamada a `ewl_dialog_new()` pasándole la posición de los botones relativa a la propia ventana. Los valores posibles son:

- EWL_POSITION_TOP
- EWL_POSITION_BOTTOM
- EWL_POSITION_LEFT
- EWL_POSITION_RIGHT

Un `Ewl_Dialog` puede opcionalmente tener dibujada una linea horizontal para separar las dos secciones del diálogo. La linea se controla con `ewl_dialog_set_has_separator()` donde 0 significa no dibujar el separador y 1 significa dibujarlo. Hay una correspondiente `ewl_dialog_get_has_separator()` que devuelve 1 si hay separador y 0 de si no lo hay.

El contenido del área principal de display de la caja es controlado por medio de la función `ewl_dialog_add_widget()`. En este caso añadimos un objeto `Ewl_Text` al dialogo.

Una vez el diálogo ha sido inicializado necesitamos crear los botones deseados. Los botones son creados llamando a `ewl_dialog_set_button()`, que creará un botón. Los parámetros son la etiqueta del botón y el código de respuesta que devolver desde el botón. Hay varias etiquetas predefinidas, incluyendo:

- EWL_STOCK_OK
- EWL_STOCK_APPLY
- EWL_STOCK_CANCEL
- EWL_STOCK_OPEN
- EWL_STOCK_SAVE
- EWL_STOCK_PAUSE
- EWL_STOCK_PLAY
- EWL_STOCK_STOP

Los códigos de respuesta predefinidos son:

- EWL_RESPONSE_OPEN
- EWL_RESPONSE_SAVE
- EWL_RESPONSE_OK
- EWL_RESPONSE_CANCEL
- EWL_RESPONSE_APPLY
- EWL_RESPONSE_PLAY
- EWL_RESPONSE_PAUSE
- EWL_RESPONSE_STOP

Una vez que los botones son creados necesitan ser añadidos al diálogo y tener un callback añadido para su estado `EWL_CALLBACK_CLICKED`.

Example 7.9. Callback de diálogo EWL

```
void dialog_clicked_cb(Ewl_Widget *w, void *event, void *data) {
```

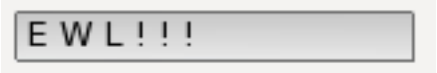
```
    int d = EWL_BUTTON_STOCK(w)->response_id;

    if (d == EWL_RESPONSE_OK)
        printf("OK\n");
    else if (d == EWL_RESPONSE_CANCEL)
        printf("CANCEL\n");

    ewl_widget_destroy(EWL_WIDGET(data));
}
```

El código de respuesta del botón que fué pulsado está disponible en el propio widget `Ewl_Button_Stock` a través de su parámetro response_id. Usando este valor podemos determinar que botón fué pulsado. También pasamos el propio `Ewl_Dialog` mediante el parámetro de datos de manera que podemos destruir el diálogo cuando hemos terminado.

ewl_entry

La caja de entrada EWL está disponible cuando necesites recibir entrada de texto del usuario. La caja funciona en simples líneas, y el callback es accionado cuando el usuario presiona la tecla "Enter".

Figure 7.5. Una caja de entrada EWL



Example 7.10. Creando una caja de entrada EWL

```
   Ewl_Widget *entry = ewl_entry_new();
   ewl_object_size_request(EWL_OBJECT(entry), 100, 15);
   ewl_object_padding_set(EWL_OBJECT(entry), 1, 1, 1, 1);
   ewl_callback_append(entry, EWL_CALLBACK_VALUE_CHANGED, entry_cb, NULL);
   ewl_widget_show(entry);
```

`Ewl_Entry` es un objeto bastante simple con el que trabajar, la única inicialización requerida es crear el nuevo objeto y vincular un callback para eventos `EWL_CALLBACK_VALUE_CHANGED`. Este ejemplo toma los pasos extra de seleccionar el tamaño con `ewl_object_size_request()` y añadir un poco de relleno al widget con `ewl_object_padding_set()`.

Example 7.11. callback de cambio de valor para Ewl_Entry

```
void entry_cb(Ewl_Widget *w, void *event, void *data) {
    char *s = ewl_entry_get_text(EWL_ENTRY(w));
    printf("%s\n", s);

    ewl_entry_set_text(EWL_ENTRY(w), "New Text");
}
```

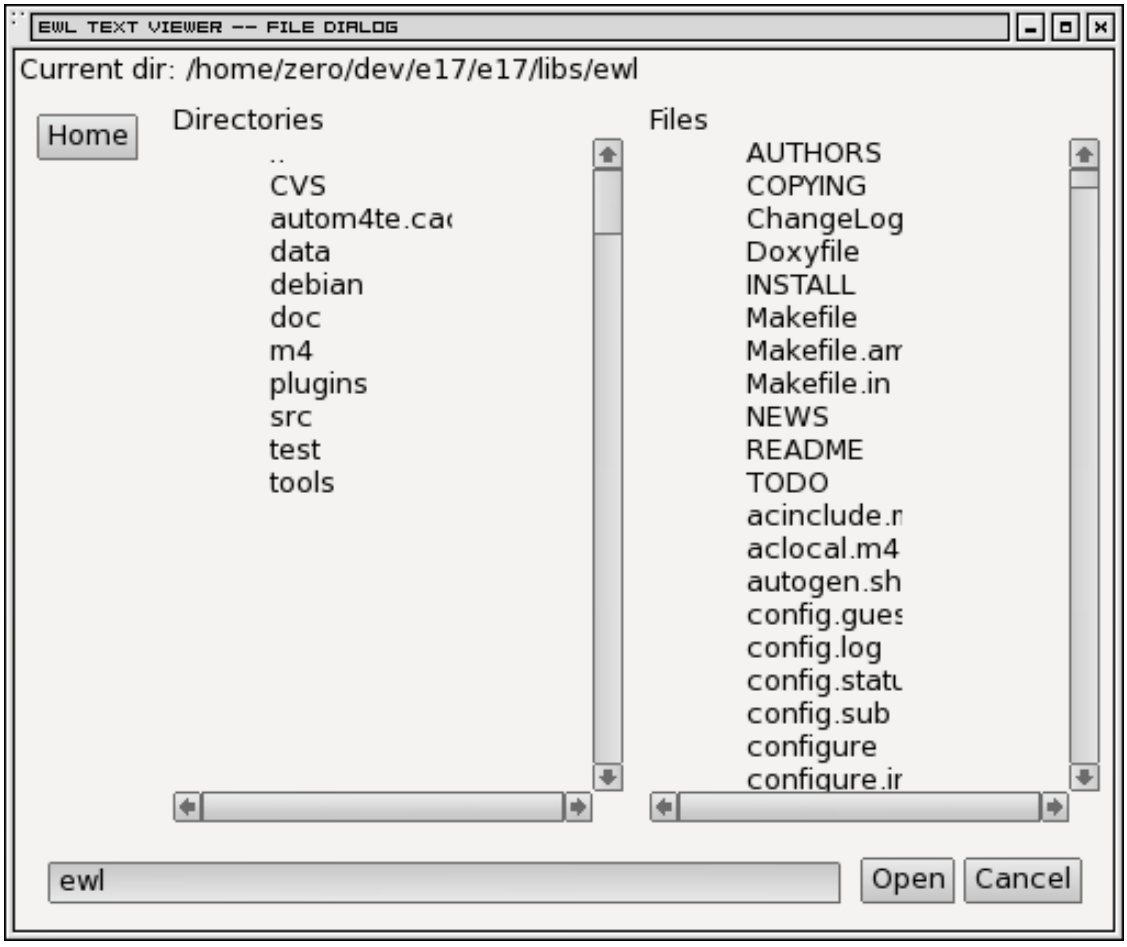
Este callback captura el valor actual del widget entrada con la llamada a `ewl_entry_get_text()` y entonces resetea el texto al valor "New Text" llamando a `ewl_entry_set_text()`.

El objeto `Ewl_Entry` te permite configurarsi el texto es o no editable con una llamada a `void ewl_entry_set_editable(Ewl_Entry *, unsigned int edit)` donde `edit` es 0 para ineditable y 1 para editable.

ewl_filedialog

A menudo se desea permitir al usuario abrir y grabar archivos. Esto puede ser fácilmente hecho con el uso de `Ewl_Filedialog`.

Figure 7.6. Un diálogo de archivo EWL



Este diálogo de archivo ha sido empotrado en su propia ventana, pero podría haber sido puesto en otra ventana de la misma manera.

Example 7.12. Creando un diálogo de archivo EWL

```
Ewl_Widget *filedialog = ewl_filedialog_new(EWL_FILEDIALOG_TYPE_OPEN);
ewl_callback_append(filedialog, EWL_CALLBACK_VALUE_CHANGED,
                    open_file_cb, NULL);
ewl_widget_show(filedialog);
```

Cuando el diálogo de archivo es creado especificas un tipo, bien `EWL_FILDIALOG_TYPE_OPEN` o bien `EWL_FILEDIALOG_TYPE_SAVE`, dependiendo del tipo de diálogo de archivo deseado. El callback `EWL_CALLBACK_VALUE_CHANGED` será ejecutado cuando el usuario pulse el botón 'Open' en el diálogo.

También es posible empaquetar otros widgets en el propio diálogo de archivo. Esto es hecho por medio de la habitual `ewl_container_child_append()`. Así, si necesitaras, por ejemplo, añadir un botón "Home", crearías el botón y lo empaquetarías en el diálogo de archivo donde aparecerá abajo en la parte izquierda.

Puedes cambiar el directorio que se ve actualmente en el diálogo de archivo ejecutando `void ewl_filedialog_set_directory(Ewl_Filedialog *, char *path)` donde `path` es la ruta completa al directorio deseado.

Example 7.13. callback para Ewl_Filedialog open

```
void open_file_cb(Ewl_Widget *w, void *event, void *data) {
    char *filename = (char *)event;
    printf("selected file %s\n", filename);
}
```

El archivo que ha sido seleccionado es pasado al callback como parámetro `event`. Si quieres quitar el diálogo de archivo puedes hacer algo similar a `ewl_widget_hide(fd_win)` donde `fd_win` es el objeto ventana que contiene el diálogo de archivo.

ewl_image

Example 7.14. Ewl_Image

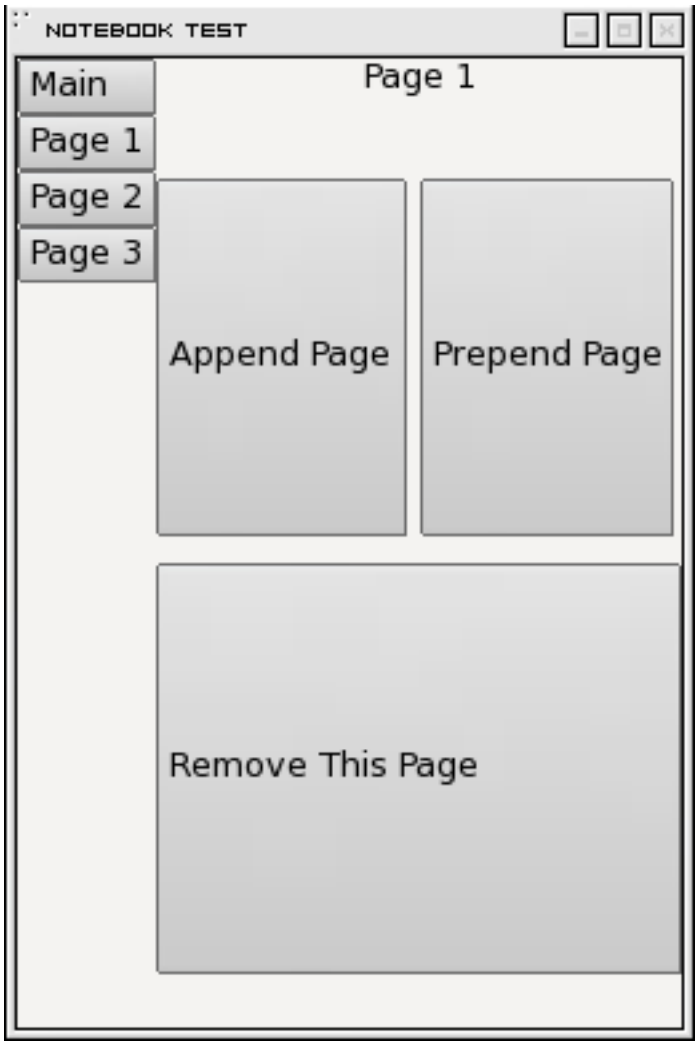
```
Ewl_Widget *i = ewl_image_new("/usr/foo/img.png", NULL);
ewl_widget_show(i);
```

La función`ewl_image_new()` toma dos parametros, la ruta a la imagen a cargar y la clave para los datos de imagen. La clave es usada principalmente para cargar grupos edje o datos con clave como imagen.

ewl_menu

ewl_notebook

Figure 7.7. Un libro de notas EWL



ewl_password

El widget `ewl_password` provee funcionalidad similar similar al widget `ewl_text` , excepto que cualquier texto tecleado no será mostrado, en su lugar un carácter oscurecedor configurable será mostrado.

Figure 7.8. Un diálogo de contraseña EWL

Example 7.15. Creando un diálogo de contraseña EWL

```
    Ewl_Widget *p = ewl_password_new("default");
    ewl_password_set_obscure(EWL_PASSWORD(p), "-");
    ewl_callback_append(p, EWL_CALLBACK_VALUE_CHANGED, passwd_cb, NULL);
    ewl_widget_show(p);
```

El carácter oscurecedor por defecto es un carácter "*". Esto se puede cambiar fácilmente llamando a `ewl_password_set_obscure(Ewl_Password *, char)`. Hay también una correspondiente función `char ewl_password_get_obscure(Ewl_Password *)` para obtener el carácter obscurecedor actual. Como con el widget `ewl_text` hay dos funciones para obtener y colocar el texto del widget: `ewl_password_set_text(Ewl_Password *, char *)` y `char *ewl_password_get_text(Ewl_Password *)`.

Cuando el usuario presiona la tecla enter en la caja de contraseña se activará un `EWL_CALLBACK_VALUE_CHANGED` will be triggered.

Example 7.16. callback de cambio de valor de Ewl_Password

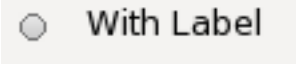
```
void passwd_cb(Ewl_Widget *, void *event, void *data) {
    char *text = ewl_password_get_text(EWL_PASSWORD(w));
    printf("text: %s\n", text);
}
```

ewl_progressbar

Figure 7.9. Una barra de progreso EWL

ewl_radiobutton

Figure 7.10. Un radiobutton EWL



ewl_scrollpane

ewl_seeker

Figure 7.11. Un buscador EWL

Example 7.17. Creando un buscador EWL

```
    Ewl_Widget *s = ewl_seeker_new(EWL_ORIENTATION_HORIZONTAL);
    ewl_seeker_value_set(EWL_SEEKER(s), 5.0);
    ewl_seeker_range_set(EWL_SEEKER(s), 10.0);
    ewl_seeker_step_set(EWL_SEEKER(s), 1);
    ewl_callback_append(s, EWL_CALLBACK_VALUE_CHANGED, seeker_cb, NULL);
    ewl_widget_show(s);
```

Example 7.18. Ewl_Seeker callback

```
void seeker_cb(Ewl_Widget *w, void *event, void *data) {
    double val = ewl_seeker_value_get(EWL_SEEKER(w));
    printf("%f\n", val);
}
```

ewl_spinner

Figure 7.12. Un spinner EWL

ewl_table

ewl_text

El widget `Ewl_Text` provee un widget de texto multilínea. Puede ser usado cuando se requiere mostrar texto en una aplicación. Funciona bien con el `Ewl_Scrollpane` para proveer un área de texto desplazable.

Example 7.19. Código Ewl_Text

```
Ewl_Widget *text = ewl_text_new("text");
ewl_widget_show(text);
```

Crear el objeto `Ewl_Text` es bastante simple, el objeto será iniciado para mostrar el parámetro a `ewl_text_new()`.

Una vez el objeto de texto es creado puedes cambiar el texto, recuperar el contenido del texto actual u obtener la longitud del texto con:

- `ewl_text_text_set(Ewl_Text *, char *)`
- `ewl_text_text_prepend(Ewl_Text *, char *)`
- `ewl_text_text_append(Ewl_Text *, char *)`
- `ewl_text_text_insert(Ewl_Text *, char *, int index)`
- `char *ewl_text_text_get(Ewl_Text *)`
- `int ewl_text_length_get(Ewl_Text *)`

El widget `Ewl_Text` te permite hacer cambios de estilización al texto en el widget. Diferentes porciones del texto pueden ser de diferentes colores, fuentes, o estilos. El estilo aplicaco a un widget se basa en que está configurado cuando el texto es añadido al widget. Así, si quieres que tu texto sea rojo, necesitas seleccionar el color del objeto `Ewl_Text` *antes*, añadiendo el texto después.

El color del texto puede ser manipulado con la llamada `ewl_text_color_set(Ewl_Text *, int r, int g, int b, int a)` , mientras que la información de color actual puede ser obtenida con `ewl_text_color_get(Ewl_Text *, int *r, int *g, int *b, int *a)`.

La configuración de fuentes del texto puede ser manipulada con `ewl_text_font_set(Ewl_Text *, char *font, int size)` . Con las llamadas para obtener el nombre de fuente actual y el tamaño definidas como: `char *ewl_text_font_get(Ewl_Text *)` and `int ewl_text_font_size_get(Ewl_Text *)`.

Para obtener o colocar la alineación del widget de texto hay dos funciones: `ewl_text_align_set(Ewl_Text *, unsigned int align)` y `unsigned int ewl_text_align_get(Ewl_Text *)`. Donde el parámetro de alineación es una de las opciones de alineación EWL:

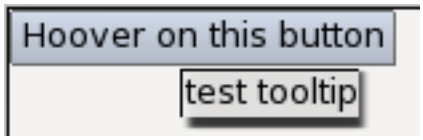
- EWL_FLAG_ALIGN_CENTER
- EWL_FLAG_ALIGN_LEFT
- EWL_FLAG_ALIGN_RIGHT
- EWL_FLAG_ALIGN_TOP
- EWL_FLAG_ALIGN_BOTTOM

También es posible colocar el estilo del texto. Esto puede incluir cosas como hacer el texto en negrita o colocar sombras suaves. Los estilos que están disponibles son obtenidos por medio de la librería Etox e incluyen actualmente:

- bold
- outline
- plain
- raised
- shadow
- soft_shadow

ewl_tooltip

Figure 7.13. Un tooltip EWL



ewl_tree

ewl_media

El widget `Ewl_Media` permite empotrar objetos de vídeo en tu aplicación. Esto es hecho mediante un envoltorio de la librería Emotion.

Figure 7.14. Un objeto media EWL

Example 7.20. Ewl_Media code

```
Ewl_Media *m = ewl_media_new(file);
ewl_callback_append(m, EWL_CALLBACK_REALIZE, video_realize_cb, NULL);
ewl_callback_append(m, EWL_CALLBACK_VALUE_CHANGED, video_change_cb, NULL);
ewl_widget_show(m);
```

Crear el objeto de vídeo básico es tan simple como crear el objeto y mostrarlo (suponiendo que lo hayas añadido a cualquier contenedor en el que esté colocado). Vinculamos los dos callbacks `EWL_CALLBACK_REALIZE` y `EWL_CALLBACK_VALUE_CHANGED`. Vinculamos el callback realize para poder determinar la longitud del vídeo a ser mostrado si lo deseamos. Esto solo está disponible después de que el vídeo haya sido realizado, y devolverá 0 hasta que haya sido realizado. El callback de cambio de valor será llamado cuando emotion avance el vídeo. Esto puede ser usado para un temporizador o una seekbar y hacer que autoavance con el vídeo.

Example 7.21. Ewl_Media callbacks

```
void video_realize_cb(Ewl_Widget *w, void *event, void *data) {
    double len = ewl_media_length_get(EWL_MEDIA(video));
}

void video_change_cb(Ewl_Widget *w, void *event, void *data) {
    char buf[512];
    int h, m;
    double s;

    ewl_media_position_time_get(EWL_MEDIA(video), &h, &m, &s);
    snprintf(buf, sizeof(buf), "%02i:%02i:%02.0f", h, m, s);
}
```

El vídeo que está siendo mostrado puede ser cambiado llamando a `ewl_media_media_set(Ewl_Media *, char *)` o si solo quieres saber lo que se está reproduciendo actualmente puedes llamar a `char *ewl_media_media_get(Ewl_Media *)`. La longitud del vídeo actual puede ser obtenida llamando a `int ewl_media_length_get(Ewl_Media *)`. La longitud puede también ser obtenida como valor de tiempo llamando a `ewl_media_length_time_get(Ewl_Media *, int h, int m, double s)`.

Puedes empezar la reproducción de vídeo pasando 1 a `ewl_media_play_set(Ewl_Media *, int)` o parar el video pasando 0 a la misma función.

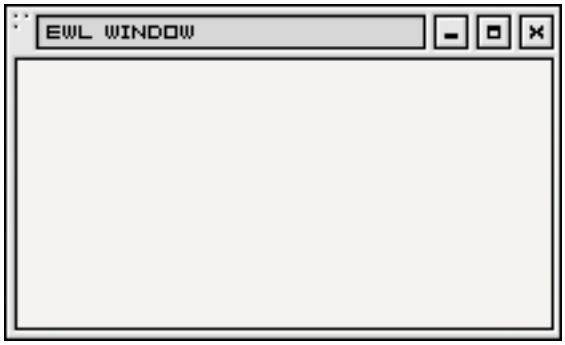
Para determinar si el codec de video permite búsqueda en el video puedes llamar a `int ewl_media_seekable_get(Ewl_Media *)` que devolverá 1 si el video permite búsqueda, y 0 si no. `double ewl_media_position_get(Ewl_Media *)` es usada para determinar la posición actual en el video, mientras que`ewl_media_position_set(Ewl_Media *, double position)` puede ser usada para seleccionar la posición en el video. Este valor puede también ser obtenido como horas, minutos, y segundos llamando a `ewl_media_position_time_get(Ewl_Media *, int h, int m, double s)`.

Si quieres cambiar la configuración de audio del video hay varias funciones disponibles. Estas incluyen la habilidad de obtener/colocar la configuración de mute: `int ewl_media_audio_mute_get(Ewl_Media *)` y `ewl_media_audio_mute_set(Ewl_Media *, int)`. También puedes obtener/colocar el volumen del video por medio de: `int ewl_media_audio_volume_get(Ewl_Media *)` y `ewl_media_audio_volume_set(Ewl_Media *, int)`.

ewl_window

Una ewl_window será usada por toda aplicación EWL. Esta es la ventana que mostrará todos los otros widgets que se deseen.

Figure 7.15. Una ventana EWL



Example 7.22. Creando una ventana EWL

```
    Ewl_Widget *window = ewl_window_new();
    ewl_window_title_set(EWL_WINDOW(window), "foo window");
    ewl_window_class_set(EWL_WINDOW(window), "foo_class");
    ewl_window_name_set(EWL_WINDOW(window), "foo_name");
    ewl_object_size_request(EWL_OBJECT(window), 300, 400);
    ewl_callback_append(window, EWL_CALLBACK_DELETE_WINDOW, win_del_cb, NULL);
    ewl_widget_show(window);
```

Iniciar la ventana básica es bastante simple. Tomamos los pasos extra de llamar: `ewl_window_title_set()`, `ewl_window_name_set()` y `ewl_window_class_set()` para rellenar la información que usa el gestor de ventanas.

Dado que la ventana es un `Ewl_Object` como cualquier otro usamos `ewl_object_size_request()` para requerir el tamaño inicial de nuestra ventana. Podríamos haber llamado también a `ewl_object_minimum_size_set()` y `ewl_object_maximum_size_set()` para restringir los tamaños mínimo y máximo de nuestra ventana.

El callback principal usado por una Ewl_Window es `EWL_CALLBACK_DELETE_WINDOW`. Este será llamado cuando la ventana sea destruida por el gestor de ventanas. Debería ser usado para limpiar los recursos que la aplicación ha usado antes de salir de la aplicación.

Example 7.23. Ewl Window destroy callback

```
void win_del_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_destroy(w);
    ewl_main_quit();
}
```

Algunas de las otras operaciones relacionadas con el objeto Ewl_Window son:

- `char *ewl_window_title_get(Ewl_Window *)`
- `char *ewl_window_name_get(Ewl_Window *)`

- `char *ewl_window_class_get(Ewl_Window *)`
- `void ewl_window_borderless_set(Ewl_Window *)`
- `void ewl_window_move(Ewl_Window *, int x, int y)`
- `void ewl_window_position_get(Ewl_Window *, int *x, int *y)`

Las primeras tres llamadas son bastante autoexplicatorias. `ewl_window_borderless_set()` puede ser usada para decir al gestor de ventanas que no muestre ninguna decoración alrededor de la ventana, esto incluye el borde y la barra de título. La función `ewl_window_move()` es usada para posicionar la ventana en un lugar específico del escritorio, indexado desde la esquina superior izquierda. También hay una función `ewl_window_position_get()` que devolverá la posición de la ventana en el escritorio.

Chapter 8. Contribuyendo

Si encontraste este documento útil, pero carente en algún respecto, considera el contribuir al propio documento. Este documento está disponible bajo una licencia abierta y cualquier contribución es muy apreciada. Contribuciones (en Inglés) pueden ser enviadas a : zero@perplexity.org [mailto:zero@perplexity.org]. Para cuestiones específicas de la traducción al castellano intenta : rtx [mailto:rtx@papelmail.com]

Observa que cualquier contribución a este documento necesita estar licenciada bajo la licencia que este documento usa, la Creative Commons NonCommercial-ShareAlike 1.0 License.

Si deseas contribuir a la EWL o a otra parte de las EFL, hecha un vistazo al website de www.enlightenment.org [http://www.enlightenment.org]. Toda la información sobre acceder CVS y las listas de email puede ser encontrada allí.

Gracias.

Appendix A. Ejemplo de reproductor de media EWL

Example A.1. Reproductor de media EWL

```
#include <Ewl.h>

static Ewl_Widget *video;
static Ewl_Widget *fd_win;
static Ewl_Widget *seeker;

typedef struct {
    char *name;
    Ewl_Callback_Function func;
} Control;

void del_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_hide(w);
    ewl_widget_destroy(w);
    ewl_main_quit();
}

void play_cb(Ewl_Widget *w, void *event, void *data ) {
    ewl_media_play_set(EWL_MEDIA(video), 1);
}

void stop_cb(Ewl_Widget *w, void *event, void *data ) {
    ewl_media_play_set(EWL_MEDIA(video), 0);
}

void ff_cb(Ewl_Widget *w, void *event, void *data ) {
    double p = ewl_media_position_get(EWL_MEDIA(video));
    ewl_media_position_set(EWL_MEDIA(video), p + 10.0);
}

void rew_cb(Ewl_Widget *w, void *event, void *data ) {
    double p = ewl_media_position_get(EWL_MEDIA(video));
    ewl_media_position_set(EWL_MEDIA(video), p - 10.0);
}

void video_realize_cb(Ewl_Widget *w, void *event, void *data) {
    double len = ewl_media_length_get(EWL_MEDIA(video));
    ewl_seeker_range_set(EWL_SEEKER(seeker), len);
}

void video_change_cb(Ewl_Widget *w, void *event, void *data) {
    char buf[512];
    int h, m;
    double s;
    Ewl_Text *t = (Ewl_Text *)data;
    double pos = ewl_media_position_get(EWL_MEDIA(video));

    ewl_seeker_value_set(EWL_SEEKER(seeker), pos);
    ewl_media_position_time_get(EWL_MEDIA(video), &h, &m, &s);
    snprintf(buf, sizeof(buf), "%02i:%02i:%02.0f", h, m, s);
    ewl_text_text_set(t, buf);
```

```
}

void seeker_move_cb(Ewl_Widget *w, void *event, void *data) {
    double val = ewl_seeker_value_get(EWL_SEEKER(seeker));
    ewl_media_position_set(EWL_MEDIA(video), val);
}

void fd_win_del_cb(Ewl_Widget *w, void *event, void *data) {
    ewl_widget_hide(w);
    ewl_widget_destroy(w);
}

void open_file_cb(Ewl_Widget *w, void *event, void *data) {
    char *file = NULL;

    ewl_widget_hide(fd_win);
    file = (char *)event;
    if (file)
        ewl_media_media_set(EWL_MEDIA(video), file);
}

void open_cb(Ewl_Widget *w, void *event, void *data ) {
    Ewl_Widget *fd = NULL;

    if (fd_win) {
        ewl_widget_show(fd_win);
        return;
    }

    fd_win = ewl_window_new();
    ewl_window_title_set(EWL_WINDOW(fd_win), "EWL Media Open");
    ewl_window_class_set(EWL_WINDOW(fd_win), "EWL_Media_Open");
    ewl_window_name_set(EWL_WINDOW(fd_win), "EWL_Media_Open");
    ewl_callback_append(fd_win, EWL_CALLBACK_DELETE_WINDOW,
                        fd_win_del_cb, NULL);
    ewl_widget_show(fd_win);

    fd = ewl_filedialog_new(EWL_FILEDIALOG_TYPE_OPEN);
    ewl_container_child_append(EWL_CONTAINER(fd_win), fd);
    ewl_callback_append(fd, EWL_CALLBACK_VALUE_CHANGED, open_file_cb, NULL);
    ewl_widget_show(fd);
}

void key_up_cb(Ewl_Widget *w, void *event, void *data) {
    Ewl_Event_Key_Up *e = (Ewl_Event_Key_Up *)event;

    if (!strcmp(e->keyname, "p"))
        ewl_media_play_set(EWL_MEDIA(video), 1);

    else if (!strcmp(e->keyname, "s"))
        ewl_media_play_set(EWL_MEDIA(video), 0);

    else if (!strcmp(e->keyname, "q"))
        del_cb(w, event, data);
}

int main(int argc, char ** argv) {
    Ewl_Widget *win = NULL, *o = NULL, *b = NULL;
    Ewl_Widget *controls = NULL, *time = NULL;
    char * file = NULL;

    if (!ewl_init(&argc, argv)) {
        printf("Can't init ewl");
        return 1;
```

```
}

if (argc > 1)
    file = argv[1];

win = ewl_window_new();
ewl_window_title_set(EWL_WINDOW(win), "EWL Media test");
ewl_window_name_set(EWL_WINDOW(win), "EWL_Media_test");
ewl_window_class_set(EWL_WINDOW(win), "EWL_Media_test");
ewl_callback_append(win, EWL_CALLBACK_DELETE_WINDOW, del_cb, NULL);
ewl_callback_append(win, EWL_CALLBACK_KEY_UP, key_up_cb, NULL);
ewl_object_size_request(EWL_OBJECT(win), 320, 280);
ewl_object_fill_policy_set(EWL_OBJECT(win), EWL_FLAG_FILL_ALL);
ewl_widget_show(win);

/* box to contain everything */
b = ewl_vbox_new();
ewl_container_append_child(EWL_CONTAINER(win), b);
ewl_object_fill_policy_set(EWL_OBJECT(b), EWL_FLAG_FILL_ALL);
ewl_widget_show(b);

/* create the time widget now so we can pass it to the video as data */
time = ewl_text_new("00:00:00");

/* the video */
video = ewl_media_new(file);
ewl_container_child_append(EWL_CONTAINER(b), video);
ewl_object_fill_policy_set(EWL_OBJECT(video), EWL_FLAG_FILL_ALL);
ewl_callback_append(video, EWL_CALLBACK_REALIZE, video_realize_cb, NULL);
ewl_callback_append(video, EWL_CALLBACK_VALUE_CHANGED, video_change_cb, time);
ewl_widget_show(video);

/* box to contain contols and scrollers */
controls = ewl_vbox_new();
ewl_object_fill_policy_set(EWL_OBJECT(controls),
            EWL_FLAG_FILL_VSHRINK | EWL_FLAG_FILL_HFILL);
ewl_container_child_append(EWL_CONTAINER(b), controls);
ewl_widget_show(controls);

/* hold he controls */
b = ewl_hbox_new();
ewl_container_child_append(EWL_CONTAINER(controls), b);
ewl_widget_show(b);

{
    Control controls [] = {
        { "play", play_cb },
        { "stop", stop_cb },
        { "rewind", rew_cb },
        { "fast forward", ff_cb },
        { "open", open_cb },
        { NULL, NULL }
    };
    int i;

    for(i = 0; controls[i].name != NULL; i++) {
        o = ewl_button_with_stock_new(controls[i].name);
        ewl_container_child_append(EWL_CONTAINER(b), o);
        ewl_callback_append(o, EWL_CALLBACK_CLICKED,
                                controls[i].func, NULL);
        ewl_widget_show(o);
    }
}
```

```
    b = ewl_hbox_new();
    ewl_container_child_append(EWL_CONTAINER(controls), b);
    ewl_widget_show(b);

    /* the video seeker */
    seeker = ewl_seeker_new(EWL_ORIENTATION_HORIZONTAL);
    ewl_container_child_append(EWL_CONTAINER(b), seeker);
    ewl_object_fill_policy_set(EWL_OBJECT(seeker),
            EWL_FLAG_FILL_VSHRINK | EWL_FLAG_FILL_HFILL);
    ewl_seeker_value_set(EWL_SEEKER(seeker), 0.0);
    ewl_seeker_range_set(EWL_SEEKER(seeker), 0.0);
    ewl_seeker_step_set(EWL_SEEKER(seeker), 1.0);
    ewl_callback_append(seeker, EWL_CALLBACK_VALUE_CHANGED, seeker_move_cb, NULL);
    ewl_widget_show(seeker);

    /* the time text spot */
    ewl_container_child_append(EWL_CONTAINER(b), time);
    ewl_object_insets_set(EWL_OBJECT(time), 0, 3, 0, 0);
    ewl_object_fill_policy_set(EWL_OBJECT(time), EWL_FLAG_FILL_SHRINK);
    ewl_widget_show(time);

    ewl_main();
    return 0;
}
```