# Guida introduttiva alle librerie grafiche GTK+

**Sara Casti - Gianmarco Cherchi - Davide Gessa**

## Breve introduzione

*GTK (GimpToolKit) è un insieme di strumenti per la creazione di interfacce grafiche inizialmente progettato per il programma di grafica multipiattaforma GIMP. GTK è scritto interamente in C ma può essere utilizzato con una vasta gamma di linguaggi di programmazione (C++, Pearl, Java, Pythonecc). In questa guida vedremo i concetti base per creare l'interfaccia grafica di un programma scritto in C utilizzando le librerie GTK su Windows e Linux.*

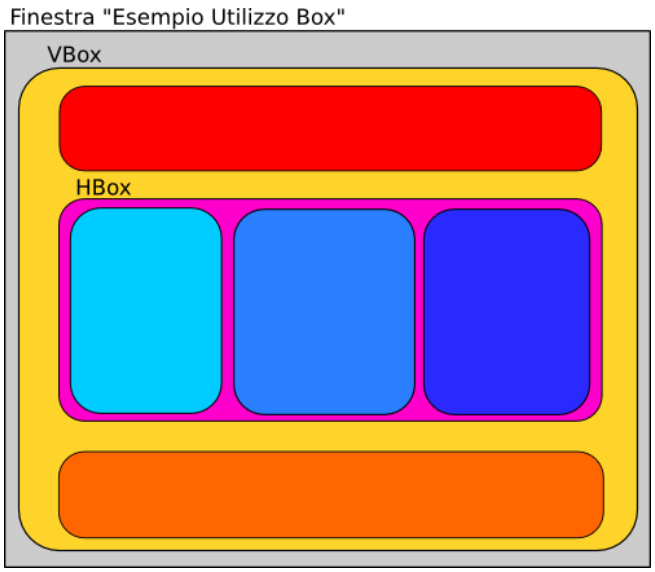
**Concetti base:**

Nelle librerie GTK gli oggetti dell'interfaccia (finestre, bottoni, etichette, immagini ecc) prendono il nome di **Widget** e hanno tutti le stesse caratteristiche base (che vedremo in seguito). Ad ogni Widget sono associati dei particolare **eventi** (clik col mouse, timer, modifica di un testo ecc) ai quali è possibile far corrispondere delle azioni (**callback**) ovvero chiamate a funzioni precedentemente implementate.

**Disposizione degli oggetti nelle finestre:**

Per disporre i widgets all'interno della finestra si possono utilizzare diverse tecniche: box, griglie, coordinate fisse, layout ecc. in questo corso tratteremo soltanto la disposizione degli oggetti tramite box.

Le box che permettono di posizionare i widgest all'interno delle finestre sono: **vertical box (vbox)** e **horizontal box (hbox)**. Tramite le vbox è possibile disporre più widgets in verticale e analogamente, tramite le hbox è possibile la disposizione orizzontale. Essendo le box a loro volta dei widgets è possibile inserire delle box all'interno di altre box.




# Installazione delle librerie e configurazione dell'IDE

## Windows

- Scaricare il bundle: http://ftp.gnome.org/pub/gnome/binaries/win32/gtk+/2.22/gtk+-bundle_2.22.1-20101227_win32.zip
- Creare la cartella C:/gtk e scompattarlo al suo interno.
- Copiare il contenuto di C:/gtk/bin (solo i file *.dll) dentro C:/Windows/System32

**Configurazione dell'IDE CodeBlocks**

- Creare un nuovo progetto
- Seleziona **gtk+** come tipo di progetto. Quando viene richiesta la cartella in cui risiede la libreria, selezionare quella in cui è stata scompattata la cartella, per esempio C:/gtk. Codeblocks dovrebbe fare in automatico la configurazione delle cartelle; in caso contrario vengono richiesti i path delle cartelle **bin**, **include** e **lib** (trascurare le altre)
- Una volta aperto il progetto fare click col tasto destro sull'icona del progetto nella barra laterale a sinistra e selezionare **Build options**. Andare su **Search directories** e premendo il bottone **Add** aggiungere la cartella **gdk-pixbuf-2.0** che si trova dentro include; Andare su **Linker settings** e premendo il bottone **Add** aggiungere le librerie **gdk-win32-2.0.lib e gdk_pixbuf-2.0.lib** che si trovano dentro la cartella lib. (questa operazione va ripetuta per ogni nuovo progetto).
- Compilare il programma di prova "main.c" che viene creato automaticamente nel progetto e mandare in esecuzione.
  ***NB**: se compare un errore in questa fase andare nella cartella bin, copiare tutti i file *.dll e incollarli nella cartella dove è contenuto l'eseguibile del programma creato. Questa opzione dovrà essere ripetuta per ogni nuovo progetto!*

**Configurazione dell' IDE Codelite**

- Creare un nuovo progetto di tipo **Simple Executable (GCC)** selezionando tra le categorie la voce **ALL**.
- Cliccare col tasto destro del mouse sul progetto, andare su **settings.** Nella sezione **compiler** aggiungere nelle opzioni la stringa **$(shell C:\gtk\bin\pkg-config --cflags gtk+-win32-2.0),** nella sezione **linker** aggiungere nelle opzioni la stringa **$(shell C:\gtk\bin\pkg-config --libs gtk+-win32-2.0)**
- È necessario andare nella cartella bin, copiare tutti i file *.dll e incollarli nella cartella dove è contenuto l'eseguibile del programma creato. Questa opzione dovrà essere ripetuta per ogni nuovo progetto!

## Linux Ubuntu

- Aprire il gestore grafico dei pacchetti ed installare il pacchetto libgtk2.0-dev oppure da terminale, digitare: **sudo apt-getinstall libgtk2.0-dev**

**Configurazione dell'IDE CodeBlocks**

- Creare un nuovo progetto e seleziona **gtk+** come tipo di progetto
- Codeblocks dovrebbe automaticamente selezionare tutte le librerie da linkare e le cartelle include, non è necessaria quindi un ulteriore configurazione

- Compilare il programma di prova “main.c” che viene creato in automaticamente nel progetto e mandare in esecuzione.

**Configurazione dell’ IDE Codelite**

- Creare un nuovo progetto di tipo **Simple Executable (GCC)** selezionando tra le categorie la voce **ALL**.
- Cliccare col tasto destro del mouse sul progetto, andare su **settings.** Nella sezione **compiler** aggiungere nelle opzioni la stringa **$(shell pkg-config --cflags gtk+-2.0),** nella sezione **linker** aggiungere nelle opzioni la stringa **$(shell pkg-config --libs gtk+-2.0)**

# Iniziamo a programmare!

**Primi passi:**

Una volta creato un nuovo progetto (vedi sopra), prima di iniziare a scrivere il nostro codice, è necessario includere la libreria **gtk/gtk.h**. Il passo immediatamente successivo è quello di inizializzare la libreria all'interno del main con la funzione **gtk_init (&argc, &argv)** passando come parametri gli argomenti della riga di comando argc e argv (gli stessi argomenti vanno messi anche come parametri del main). Dopo aver creato la finestra con i relativi widgets è necessario avviare il loop del rendering tramite la funzione **gtk_main ()**. La funzione gtk_main ha il compito di visualizzare i widgets presenti nella finestra, catturare gli eventi ed eseguire le operazioni ad essi associati.

```
/// DIRETTIVE DI PREPROCESSING
#include <gtk/gtk.h>

int main(int argc, char *argv[])
{
	/// INIZIALIZZAZIONE DELLA LIBRERIA
	gtk_init (&argc, &argv);

	/// CODICE DI CREAZIONE DELLE FINESTRE E DEI WIDGET
	/// …

	/// INIZIO DEL LOOP DI RENDERING
	gtk_main();

	return 0;
}
```

Ora siamo pronti per imparare ad usare i principali **Widgets** (oggetti) della libreria gtk:

- Finestre
- Bottoni
- Input box
- Etichette
- Immagini
- Menù (a tendina)
- Contenitori a dimensioni variabili
- Elenchi
- Finestre predefinite

# Widgets

Ogni widget prima di essere utilizzato va opportunamente dichiarato (come tutte le variabili del nostro programma). I widgets sono puntatori di tipo GtkWidget inizializzati a NULL.

*Esempio:* GtkWidget *oggetto = NULL;

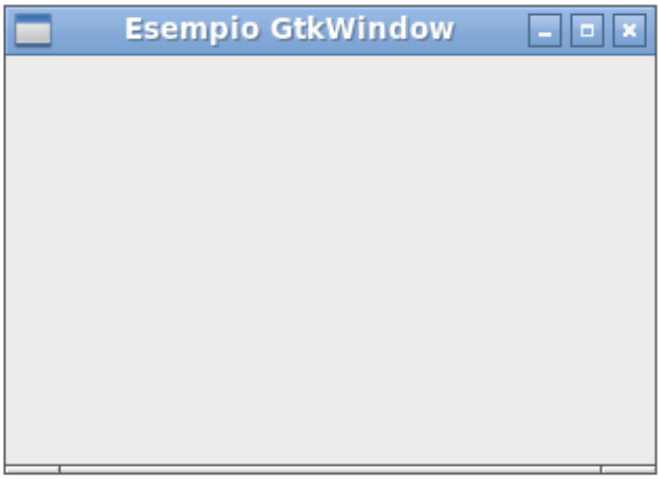
Per assegnare un'azione ad un evento di un widget si utilizza la funzione **g_signal_connect (<widget> , <evento> , G_CALLBACK( <funzione da associare>) , <argomento_funzione>).** Il parametro <evento> è una stringa che identifica il tipo di evento alla quale associare l'azione (ogni widget ha i suoi eventi); il parametro <funzione da associare> è una funzione precedentemente definita che viene associata all'evento, costruita nel seguente modo: **void nome_funzione (GtkWidget *oggetto_chiamante, altro_argomento) {… codice…}**

***NB: E' possibile passare alla funzione un unico argomento!*** *Può essere necessario passare alla funzione più di un argomento (ad esempio più widget): un modo per ovviare questo problema è quello di realizzare una struttura nella quale inserire i parametri da passare alla funzione. Alla funzione dovrà essere passato il puntatore alla struttura.*

*Esempio:* se vogliamo fare in modo che premendo un bottone venga chiamata la funzione "Hello_World()" scriveremo **void Hello_World(GtkWidget *button){… codice…}** **g_signal_connect(button, "clicked", G_CALLBACK (Hello_World), NULL);** dove "clicked" è l'evento associato al click del mouse.

Qualsiasi widget può essere "distrutto" con la funzione **gtk_widget_destroy(<puntatore_widget>).**

## Windows (finestre)




**Creazione:**

Per la creazione della finestra si utilizza la funzione **gtk_window_new(<tipo finestra>)** che ha come unico parametro il tipo di finestra da creare. I possibili valori del parametro sono:

- GTK_WINDOW_TOPLEVEL : finestra normale
- GTK_WINDOW_POPUP : finestra a comparsa

*Esempio:* `GtkWidget *finestra = gtk_window_new(GTK_WINDOW_TOPLEVEL);`

**Funzioni:**

È possibile modificare le proprietà della finestra con varie funzioni. Le più importanti sono:

- **gtk_window_set_title(GTK_WINDOW(<puntatore finestra>) , <titolo> )** per modificare il titolo della finestra. Il parametro <titolo> è una stringa che contiene il nome della finestra.
- **gtk_window_set_default_size(GTK_WINDOW(<puntatore_finestra>) , <larghezza> , <altezza>)** per impostare la dimensione iniziale della finestra. I parametri <larghezza> e <altezza> sono dei valori interi che rappresentano le dimensioni della finestra espresse in pixel.
- **gtk_window_set_position (GTK_WINDOW (<puntatore_finestra>) , <posizione>)** per modificare la posizione iniziale della finestra. I possibili valori del parametro <posizione> sono:
  - GTK_WIN_POS_NONE (nessuna posizione)
  - GTK_WIN_POS_CENTER (posizione centrale)
  - GTK_WIN_POS_CENTER_ON_PARENT (centrata rispetto alla finestra principale)
- **gtk_container_set_border_width (GTK_CONTAINER (<puntatore_finestra>) , <dimensione_bordo>)** per impostare la distanza tra i widget della finestra e il bordo. Il parametro <dimensione_bordo> è un valore intero espresso in pixel.

**Hbox e Vbox**

Ultimata la creazione della finestra, per disporre gli oggetti al suo interno è necessario dividerla in hbox e vbox. Le box vanno precedentemente dichiarate come un normale widget e vanno create con la funzione **gtk_vbox_new (<omogeneo> , <spazio>)** (rispettivamente gtk_hbox_new (…)per le hbox ). Il parametro <omogeno> è un booleano che indica se gli oggetti contenuti nella box hanno una dimensione omogenea tra di loro; il parametro <spazio> è un valore intero che indica la distanza tra gli oggetti della stessa box espressa in pixel.

*Esempio:* `GtkWidget *vbox1 = gtk_vbox_new (TRUE, 100);`

Box e finestre vanno immaginate come dei contenitori all'interno dei quali è possibile inserire i widgets.

- Per inserire una box all'interno di una finestra si utilizza la funzione **gtk_container_add (GTK_CONTAINER (<finestra_contenitore>) , <box>)**

- Per inserire un widget qualsiasi all'interno di una box si utilizza la funzione **gtk_box_pack_start (GTK_BOX (<box_contenitore>) , <widget> , <espansione_widget> , <riempimento_totale> , <distanza>)** . Il parametro <espansione_widget> è un booleano che indica se il widget considerato deve espandersi per tutta la larghezza della box (in caso di hbox) o per tutta l'altezza (nel caso di vbox); il parametro <riempimento_totale> è un booleano che indica se il widget considerato deve espandersi per tutta la dimensione della box (in altezza e larghezza); il parametro <distanza> è un intero che indica la distanza tra i widgets della stessa box espressa in pixel.

**Eventi:**

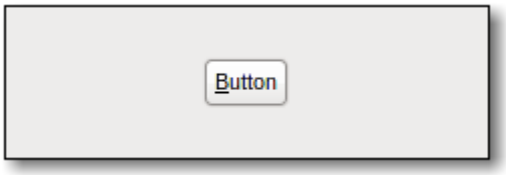
Gli eventi relativi al Widget di tipo finestra sono diversi ma nel nostro caso ci basta conoscerne uno: "**destroy"** evento relativo al click della X di chiusura della finestra. In genere questo evento viene associato alla funzione predefinita **gtk_main_quit** che termina l'esecuzione del programma.

*Esempio:* `g_signal_connect(finestra, "destroy", gtk_main_quit, NULL);`

Una volta create la finestra, e gli eventuali widget contenuti al suo interno (che vedremo dopo uno per uno), è necessario chiamare la funzione **gtk_widget_show_all (<puntatore_finestra>)** per visualizzarli.

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

int main(int argc, char *argv[])
{
    ///DICHIARAZIONE DEL WIDGET FINESTRA
    GtkWidget *finestra = NULL;
    /// DICHIARAZIONE DELLE BOX
    GtkWidget *vbox = NULL;
    GtkWidget *hbox1 = NULL;
    GtkWidget *hbox2 = NULL;

    gtk_init(&argc, &argv);

    /// CREAZIONE DELLA FINESTRA
    finestra = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_container_set_border_width(GTK_CONTAINER (finestra), 8);
    gtk_window_set_title(GTK_WINDOW(finestra), "Titolo");
    gtk_window_set_position(GTK_WINDOW(finestra), GTK_WIN_POS_CENTER);
    gtk_window_set_default size(GTK_WINDOW(finestra), 500, 600);
    /// ASSOCIA AL CLICK DELLA X LA CHIUSURA DEL PROGRAMMA
    g_signal_connect(finestra, "destroy", gtk_main_quit, NULL);

    /// CREAZIONE DI UNA VBOX
    vbox = gtk_vbox_new(TRUE, 6);
    gtk_container_add(GTK_CONTAINER (finestra), vbox);
    /// CREAZIONE E INSERIMENTO DI HBOX
    hbox1 = gtk_hbox_new(TRUE, 6);
    gtk_box_pack_start(GTK_BOX(vbox), hbox1, TRUE, TRUE, 10);

    hbox2 = gtk_hbox_new(TRUE, 6);
    gtk_box_pack_start(GTK_BOX(vbox), hbox2, TRUE, TRUE, 10);

    /// CODICE DI CREAZIONE DEI WIDGET DELLA FINESTRA
    /// ....

    ///VISUALIZZAZIONE DELLA FINESTRA E DEL SUO CONTENUTO
    gtk_widget_show_all(finestra);

    gtk_main();
    return 0;
}
```

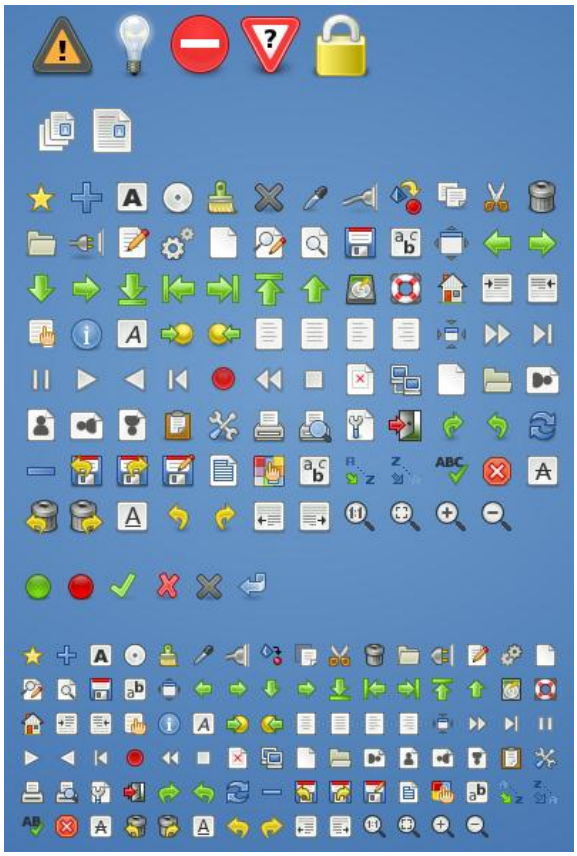
## Buttons (bottoni)




I bottoni sono dei widgets che generano un evento quando vengono cliccati. Ne esistono diversi tipi ma in questa guida vedremo soltanto quelli classici.

**Creazione**

- **GtkWidget *button1 = gtk_button_new_with_label (<label_bottone>)** per creare un bottone con testo. Il parametro <label_bottone> è una stringa che rappresenta il testo da scrivere sul bottone.
- **GtkWidget *button2 = gtk_button_new_from_stock(<ID_immagine>)** per creare un bottone con immagine. Il parametro <ID_immagine> è il codice numerico di una delle 105 immagini predefinite definite da delle costanti. L'elenco delle immagini predefinite è disponibile all'indirizzo: http://developer.gnome.org/gtk3/stable/gtk3-Stock-Items.html



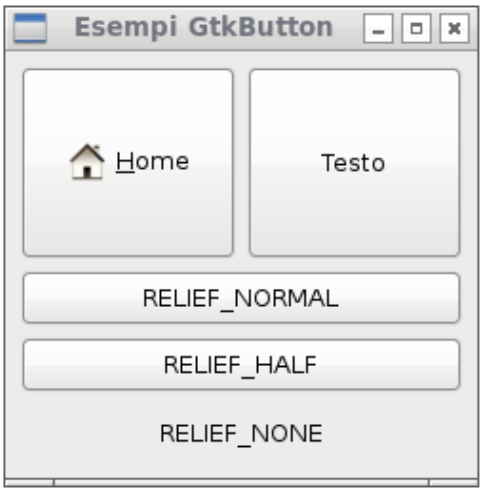
**Funzioni**

- **gtk_button_set_label(GTK_BUTTON(<puntatore_bottone>), <stringa>)** per modificare il testo all'interno del bottone. Il parametro <stringa> è la nuova stringa che si intende mettere nel bottone.
- **gtk_button_set_relief(GTK_BUTTON(<puntatore_bottone>), <stile_bottone>)** per modificare lo stile del bottone. Il parametro <stile_bottone> può essere:
  - GTK_RELIEF_NORMAL
  - GTK_RELIEF_HALF
  - GTK_RELIEF_NONE

**Eventi**

- **"clicked"** : evento generato al click del bottone

```
/// BOTTONE CON IMMAGINE
button1 = gtk_button_new_from_stock(GTK_STOCK_HOME);
g_signal_connect(G_OBJECT (button1), "clicked", G_CALLBACK (helloWorld), (gpointer) win);
gtk_box_pack_start(GTK_BOX (hbox), button1, TRUE, TRUE, 0);

/// BOTTONE CON TESTO
button2 = gtk_button_new_with_label("Testo");
g_signal_connect (G_OBJECT (button2), "clicked", G_CALLBACK (helloWorld), (gpointer) win);
gtk_box_pack_start(GTK_BOX (hbox), button2, TRUE, TRUE, 0);
```



## Labels (etichette)


Le etichette sono dei widget che permettono di mostrare a video una stringa di testo.

**Creazione**

- **gtk_label_new(<stringa>)** dove il parametro <stringa> è il testo la stampare.

**Funzioni**

- **gtk_label_set_text (GTK_LABEL(<puntatore_label>), <stringa>)** per modificare il testo della label. Il parametro <stringa> indica il nuovo testo da stampare.
- **gtk_label_set_angle(GTK_LABEL(<puntatore_label>), <angolo>)** per ruotare il testo della label di <angolo> gradi. Il parametro <angolo> è un double che rappresenta l'angolo in gradi.

```
/// LABEL DIAGONALE
GtkWidget *label1 = gtk_label_new("Ruotato diagonale");
gtk_label_set_angle(GTK_LABEL(label1), 45.);
gtk_box_pack_start(GTK_BOX(hbox), label1, TRUE, TRUE, 0);

/// LABEL ORIZZONTALE
GtkWidget *label2 = gtk_label_new("Label orizzontale");
gtk_box_pack_start(GTK_BOX(hbox), label2, TRUE, TRUE, 0);

/// LABEL VERTICALE
GtkWidget *label3 = gtk_label_new("Label verticale");
gtk_label_set_angle(GTK_LABEL(label3), 90.);
gtk_box_pack_start(GTK_BOX(hbox), label3, TRUE, TRUE, 0);
```

## Entry (input box)


Gli input box sono dei widget che permettono di acquisire una riga di testo.

**Creazione**

- **gtk_entry_new()**

**Funzioni**

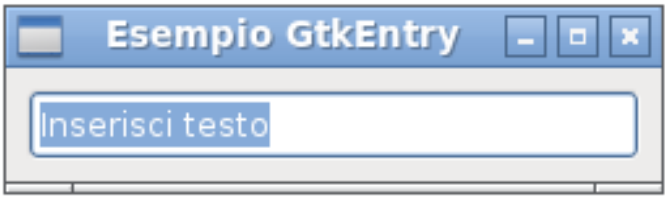
- **gtk_entry_set_text(GTK_ENTRY(<puntatore_entry>), <stringa>)** per impostare il testo dentro l'entry. Il parametro <stringa> è il testo da inserire.
- **gtk_entry_get_text(GTK_ENTRY(<puntatore_entry>))** per acquisire il testo inserito nell'entry. Questa funzione restituisce un puntatore a char.
    - Esempio: `char *testo = gtk_entry_get_text(GTK_ENTRY(entry));`

    ***NB:*** *se si vogliono acquisire dei valori interi è necessario convertire la stringa in input. Per fare questo si può utilizzare la funzione sscanf (della libreria stdlib.h) nel seguente modo:*

    ```
    char *string = gtk_entry_get_text(GTK_ENTRY(entry));
    int num;
    sscanf(string, "%d", &num);
    ```

- **gtk_entry_set_max_length(GTK_ENTRY(<puntatore_entry>) , <dimensione>)** per impostare la lunghezza massima della stringa da acquisire. Il parametro <dimensione> è un intero che rappresenta il numero di caratteri massimo della stringa.

```
GtkWidget *entry = gtk_entry_new();
gtk_entry_set_text(GTK_ENTRY(entry), "Inserisci testo");
gtk_box_pack_start(GTK_BOX(hbox), entry, TRUE, TRUE, 0);
```



## Images (immagini)


I widgets di tipo image permettono di inserire nella finestra del programma immagini di diversi formati (jpeg, png, bmp, svg ecc).

***NB**: se si utilizza l'IDE Codeblocks le immagini ( eventualmente la cartella che le contiene ) vanno posizionate allo stesso livello della cartella contenente l'eseguibile, mentre se si utilizza l'IDE Codelite vanno posizionate all'interno della cartella contenente l'eseguibile.*

**Creazione**

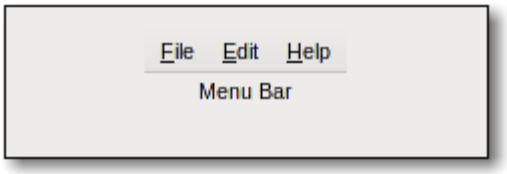
- **gtk_image_new_from_file(<nome_immagine>).**

**Funzioni**

- **gtk_image_set_from_file(GTK_IMAGE(<puntatore_immagine>) , <nome_immagine>)** per sostituire un'immagine precedentemente impostata. Il parametro <nome_immagine> è il nome della nuova immagine.
- **gtk_image_clear(GTK_IMAGE(<puntatore_immagine>))** per eliminare un'immagine dalla finestra.

```
GtkWidget *image = gtk_image_new_from_file("image.jpg");
gtk_box_pack_start(GTK_BOX(hbox), image, TRUE, TRUE, 0);
```

## Menu




Per ottenere un menù (di quelli classici a tendina che conosciamo tutti) occorre conoscere tre widgets differenti:

- **Menu Item**: widget che rappresenta un elemento cliccabile del menu (i classici “file”, “salva”, “stampa”,“esci” ecc.). Possono essere inseriti sia nel Menu Bar che nel Menu.
- **Menu Bar**: widget che rappresenta la barra del menu.
- **Menu**: widget che contiene i menu items del sottomenu (la classica tendina dei menu).

### Creazione di un menu

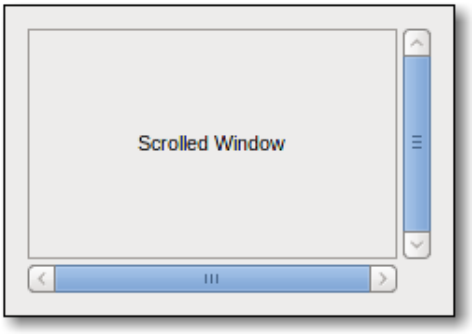
Per creare un menu è necessario utilizzare tutti widget elencati sopra nel seguente modo:

1. Creare la Menu Bar con la funzione **gtk_menu_bar_new();**
2. Creare i vari Menu Items da inserire nella barra con la funzione **gtk_menu_item_new_with_label(<stringa>)**. Il parametro <stringa> è il nome dell’ Item.
3. Per inserire gli Items creati nella barra si utilizza la funzione **gtk_menu_bar_append(GTK_MENU_BAR(<puntatore_barra>) , <puntatore_item>).** Questa funzione va ripetuta per ogni item da inserire nella barra.
4. Se si vogliono creare dei sottomenu, per ogni sottomenu creare un widget Menu (che conterrà gli Items del sottomenu) con la funzione **gtk_menu_new()**.
5. Dopo aver creato i nuovi Items, per inserirli nel sottomenu si utilizza la funzione **gtk_menu_append(GTK_MENU(<puntatore_sottomenu>) , <item>)**.
6. per abbinare i sottomenu creati agli Items della barra principale si utilizza la funzione **gtk_menu_item_set_submenu(GTK_MENU_ITEM(<puntatore_item>) , <puntatore_sottomenu>)**. Il paramentro <puntatore_item> fa riferimento ad uno degli Items della barra principale.
7. Una volta creato il menu completo si aggiunge il widget che rappresenta la barra principale in una box per posizionarlo nella finestra.

*NB: il menu così creato non è di nessun utilizzo. Bisogna collegare ad ogni Item una funzione in modo tale che, cliccandolo con il mouse, venga eseguita. Per fare questo si utilizza lo stesso metodo visto con i bottoni. L’evento da utilizzare è **“activate”**.*


```
/// CREA LA MENU BAR E LA INSERISCO NELLA VBOX PRINCIPALE
menu_bar = gtk_menu_bar_new();
gtk_box_pack_start(GTK_BOX(vbox),menu_bar, FALSE, FALSE,0);

/// CREARE IL MENU FILE
menu_file = gtk_menu_item_new_with_label("File");
gtk_menu_bar_append(GTK_MENU_BAR(menu_bar), menu_file);

/// CREARE UN SOTTOMENU', E COLLEGARLO AL MENU "File" APPENA CREATO
menu_sub = gtk_menu_new();
gtk_menu_item_set_submenu(GTK_MENU_ITEM(menu_file), menu_sub);

/// CREARE UN MENU ITEM
menu_item = gtk_menu_item_new_with_label ("Apri");

/// COLLEGARE IL MENU ITEM AD UNA CALLBACK
g_signal_connect(G_OBJECT(menu_item), "activate", G_CALLBACK(test), NULL);

/// AGGIUNGERE IL MENU ITEM "Apri" AL SOTTOMENU "Apri"
gtk_menu_append(GTK_MENU(menu_sub), menu_item);
```

## Scrolled windows (finestre di dimensioni variabili)




I widgets Scrolled Window permettono di gestire la visualizzazione di un oggetto con dimensioni maggiori rispetto a quelle della finestra in cui è contenuto, grazie all'utilizzo delle barre di scorrimento laterali. In una Scrolled Window è possibile inserire un solo oggetto! Per inserire più oggetti è necessario utilizzare una box che li contenga.

### Creazione

- **gtk_scrolled_window_new(<adattamento_orizzontale> , <adattamento_verticale>)** dove i parametri <adattamento_orizzontale> e <adattamento_verticale> rappresentano i valori di adattamento alla finestra. Impostandoli entrambi a NULL gtk provvede in automatico ad assegnare i valori adatti.

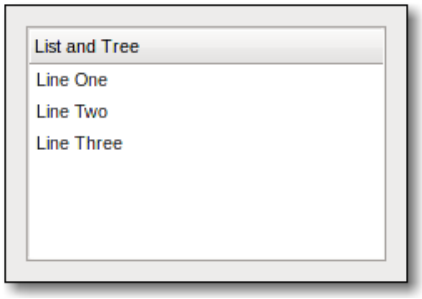
### Funzioni

- **gtk_scrolled_window_add_with_viewport(GTK_SCROLLED_WINDOW(<puntatore_scrolled>) , <widget_da_aggiungere>)** per aggiungere un oggetto all'interno della scrolled window.

*NB: la Scrolled Window va poi aggiunta in una box come tutti i widget*

```
/// CREA LA SCROLLED WINDOW
GtkWidget *scroll_box = gtk_scrolled_window_new(NULL, NULL);

/// AGGIUNGE IL WIDGET IMAGE NELLA SCROLLED WINDOW
gtk_scrolled_window_add_with_viewport(GTK_SCROLLED_WINDOW(scroll_box), image);

/// AGGIUNGE LA SCROLLED WINDOW NELLA BOX CHE VOGLIAMO
gtk_box_pack_start(GTK_BOX(vbox), scroll_box, TRUE, TRUE, 0);
```

## List (elenchi)




Gli elenchi permettono di creare una lista di oggetti il cui numero o dimensioni sono variabili. Ad esempio, la gestione di una rubrica telefonica che richiede l'aggiornamento costante del numero degli elementi. Per fare questo si utilizzano due widget:

- **List Store**: per inserire, contenere e modificare i dati dell'elenco.
- **Tree View**: per visualizzare una List Store.

**Creazione di un elenco**

1. Si crea una List Store con la funzione **gtk_list_store_new(<numero_colonne> , <tipo_di_dato_per_ogni_colonna>)**. Il parametro <tipo_di_dato_per_ogni_colonna> non si riferisce ad un solo parametro; indica per ogni colonna il tipo di dato da inserire. I tipi disponibili più importanti sono:
   - GDK_TYPE_PIXBUF : per l'inserimento di immagini
   - G_TYPE_INT : per l'inserimento di interi
   - G_TYPE_FLOAT: per l'inserimento di reali
   - G_TYPE_STRING: per l'inserimento di stringhe

   *Esempio*: `GtkListStore *list = gtk_list_store_new(2, GDK_TYPE_PIXBUF, G_TYPE_STRING);`
2. Si crea la Tree View con la funzione **gtk_tree_view_new_with_model(GTK_TREE_MODEL(<puntatore_listStore))**
3. Si creano le colonne dell'elenco, precedentemente allocate nel punto 1, con la funzione **GtkTreeViewColumn *gtk_tree_view_column_new_with_attributes(<nome_colonna> , <funzione_di_rendering> , <tipo_di_dato> , <numero_colonna> , NULL)**. Le colonne sono di tipo GtkTreeViewColumn. Il parametro <nome_colonna> è una stringa che rappresenta il nome che verrà visualizzato nella tree view. Il parametro <funzione_di_rendering> è la funzione che permette la visualizzazione della cella:
   - **gtk_celle_renderer_pixbuf_new()** : per le immagini
   - **gtk_celle_renderer_text_new()** : per testi, interi e float

   Il parametro <tipo_di_dato> rappresenta il tipo di oggetto che si inserisce nella colonna:
   - **pixbuf** : per le immagini
   - **text** : per testi, interi e float
4. Una volta create le colonne si aggiungo alla Tree View con la funzione **gtk_tree_view_append_column(GTK_TREE_VIEW(<puntatore_treeView>) , <colonna_da_aggiungere>)**
5. Una volta creato l'elenco si inserisce la Tree View nella box per posizionarlo nella finestra.

```
GtkWidget *tree_view;
GtkListStore *list_store;
GtkTreeViewColumn *column;

/// CREAZIONE DELLA LIST STORE
list_store = gtk_list_store_new(2, GDK_TYPE_PIXBUF, G_TYPE_STRING);

/// CREAZIONE DELLA TREE VIEW
tree_view = gtk_tree_view_new_with_model(GTK_TREE_MODEL(list_store));
```

```
/// CREAZIONE DI UNA COLONNA CON IMMAGINE
column = gtk_tree_view_column_new_with_attributes ("Immagine", gtk_cell_renderer_pixbuf_new (),
                                                   "pixbuf", 0, NULL);
gtk_tree_view_append_column (GTK_TREE_VIEW (tree_view), column);

/// CREAZIONE DI UNA COLONNA CON TESTO
column = gtk_tree_view_column_new_with_attributes ("Testo", gtk_cell_renderer_text_new (),
                                                   "text", 1, NULL);
gtk_tree_view_append_column (GTK_TREE_VIEW (tree_view), column);

/// INSERIMENTO DELLA TREE VIEW IN UNA VBOX
gtk_box_pack_start(GTK_BOX(vbox), tree_view, TRUE, TRUE, 0);
```

### Riempimento delle righe dell'elenco

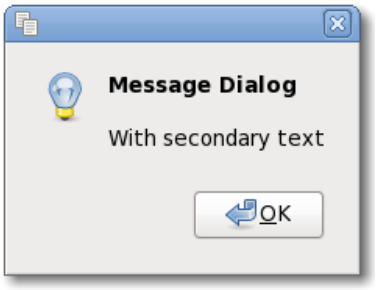
1. Si definisce una variabile del tipo **GtkTreeIter** per memorizzare la posizione della riga da creare
2. Si crea una riga vuota con la funzione **gtk_list_store_append(<puntatore_listStore> , <indirizzo_variabile_iteratore>)**
3. Si modifica il contenuto della riga creata con la funzione gtk_list_store_set(<puntatore_listStore> , <indirizzo_variabile_iteratore> , <numero_colonna> , <dato_da_inserire> , … , -1) . Il parametro

   <dato_da_inserire> nel caso di un'immagine di tipo GtkImage sarà: **gtk_image_get_pixbuf(<puntatore_GtkImage>)** . Dove ci sono i punti si inseriscono i numeri di colonna e i dati relativi a ciascuna di esse. L'ultimo parametro della funzione deve essere sempre -1 per far capire a gtk che è stato inserito l'ultimo parametro.

***NB**: ogni qualvolta viene aggiunta una riga alla List Store viene aggiornata in automatico la visualizzazione degli elementi.*

```
GtkTreeIter it;
gtk_list_store_append(list_store, &it);
gtk_list_store_set(list_store, &it,
                   0, gtk_image_get_pixbuf(image),
                   1, "Testo di prova",
                   -1);
```

## Message Dialog (finestre predefinite)




Le Message Dialog sono delle finestre predefinite della libreria gtk. Permettono di visualizzare una finestra con un titolo, un'icona, un testo principale, un testo secondario e un bottone.

**Creazione**

- **gtk_message_dialog_new(GTK_WINDOW(<puntatore_finestra_madre>) , <chiusura> , <tipo_finestra> , <tasto> , <testo_principale>) .** Il parametro <chiusura> può essere:
  - GTK_DIALOG_DESTROY_WITH_PARENT : la finestra viene "distrutta" alla chiusura della finestra madre;
  - GTK_DIALOG_MODAL : la finestra non viene "distrutta" alla chiusura della finestra madre;

  Il parametro <tipo_finestra> determina l'icona da visualizzare. Può essere dei seguenti tipi:

  - GTK_MESSAGE_INFO : finestra per informazioni
  - GTK_MESSAGE_WARNING : finestra per un messaggio di attenzione
  - GTK_MESSAGE_ERROR : finestra per un messaggio di errore

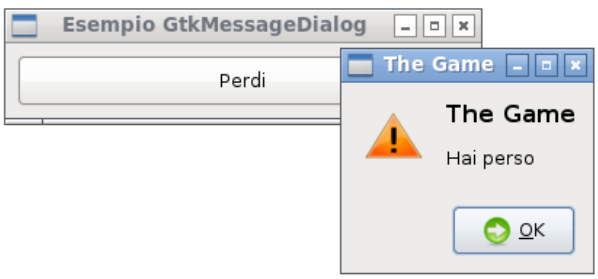
  Il parametro <tasto> rappresenta il bottone presente nella finestra:

  - GTK_BUTTONS_NONE
  - GTK_BUTTONS_OK
  - GTK_BUTTONS_CLOSE

**Funzioni**

- **gtk_message_dialog_format_secondary_text(GTK_MESSAGE_DIALOG(<puntatore_finestra>) , <testo> )** per settare il testo secondario della finestra.
- **gtk_window_set_title(GTK_WINDOW(<puntatore finestra>) , <titolo> )** per modificare il titolo della finestra.
- **gtk_dialog_run(GTK_DIALOG(<puntatore_finestra>))** per visualizzare la finestra. Questa funzione rimane in esecuzione finchè non viene chiusa la finestra.
- **gtk_widget_destroy(<puntatore_finestra>)** per "distruggere" la finestra dopo essere stata chiusa. NB: *Questa funzione va usata subito dopo la gtk_dialog_run*.

```
GtkWidget *d = gtk_message_dialog_new(win, GTK_DIALOG_DESTROY_WITH_PARENT, GTK_MESSAGE_WARNING,
                                      GTK_BUTTONS_OK, "The Game");

gtk_message_dialog_format_secondary_text(GTK_MESSAGE_DIALOG(d), "Hai perso");
gtk_window_set_title(GTK_WINDOW(d), "The Game");
gtk_dialog_run(GTK_DIALOG(d));
gtk_widget_destroy(d);
```



# Curiosità e altre cose utili

## Timer

Permette di eseguire una funzione dopo un certo numero di secondi scelti dal programmatore.
**g_timeout_add(<intervallo> , (GSourceFunc)<funzione_da_chiamare> , <parametro_funzione>)**.

Parametri:

- <intervallo> : intervallo espresso in millisecondi
- <funzione_da_chiamare> è una funzione che deve essere implementata in modo che restituisca un intero:
    - Restituisce 1 per mantenere il timer attivo e quindi richiamare nuovamente la funzione
    - Restituisce 0 per interrompere l'esecuzione del timer.
- <parametro_funzione> : è il parametro della funzione da chiamare. Se non presente si imposta a NULL

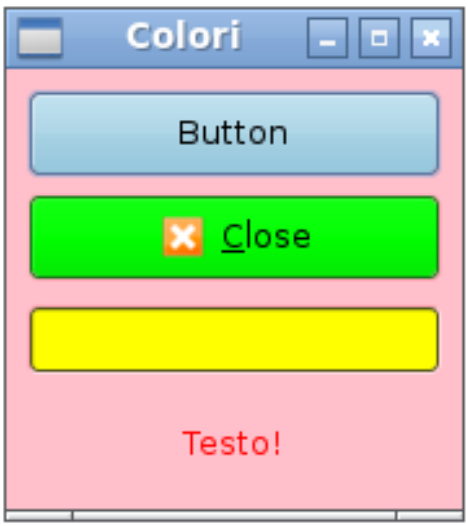
***NB**: il widget Timer è ASINCRONO ovvero è possibile effettuare altre operazioni durante la sua esecuzione*

## Cambiare il colore dei widgets

Attraverso queste tre funzioni è possibile cambiare il colore dei widgets visti nei paragrafi precedenti. In gtk i colori sono definiti da una struttura chiamata GdkColor.

- **gdk_color_parse(<colore> , <puntatore_GdkColor>)** per riempire la struttura GdkColor con il codice del colore selezionato. Il parametro colore è una stringa che contiene il nome del colore.
- **gtk_widget_modify_fg(<puntatore_widget> , GTK_STATE_NORMAL, <puntatore_GdkColor>)** per modificare il colore di foreground dell'oggetto.
- **gtk_widget_modify_bg(<puntatore_widget> , GTK_STATE_NORMAL, <puntatore_GdkColor>)** per modificare il colore di background dell'oggetto.
- **gtk_widget_modify_base(<puntatore_widget> , GTK_STATE_NORMAL, <puntatore_GdkColor>)** per modificare il colore di base dell'oggetto.

```
///DEFINISCE UNA VARIABILE DI TIPO GdkColor PER CONTENERE IL COLORE CHE USEREMO
GdkColor color;

/// CREO IL BOTTONE
button = gtk_button_new_from_stock (GTK_STOCK_CLOSE);

/// RIEMPIO LA STRUTTURA color CON IL COLORE VERDE
gdk_color_parse ("green", &color);

/// IMPOSTO IL COLORE DI SFONDO DEL WIDGET
gtk_widget_modify_bg (button, GTK_STATE_NORMAL, &color);
```



## Collegare una funzione ad un' immagine

Può essere utile collegare una funzione ad una GtkImage, in modo tale che cliccando sull'immagine questa si comporti come se fosse un GtkButton. Per fare questo i passi da seguire sono:

1. Si crea un widget **event_box** ovvero un contenitore all'interno del quale si inserisce l'immagine. L'event_box è un particolare widget a cui è possibile assegnare la maggior parte di eventi del gtk (non si può assegnare ad esempio l'evento "clicked" direttamente ad un immagine). Per creare un event box si utilizza la funzione **GtkWidget *event_box1 = gtk_event_box_new().**
2. Si inserisce l'immagine all'interno dell'event_box tramite la funzione **gtk_container_add** spiegata per le finestre.
3. Si collega l'evento **button_press_event** dell'event_box con la funzione da richiamare tramite la funzione **g_signal_connect** già vista per i bottoni.

```
/// CARICO L'IMMAGINE
image = gtk_image_new_from_file ("pig1.png");

/// CREO L'EVENT BOX
event_box = gtk_event_box_new();

/// AGGIUNGO L'IMMAGINE NELL'EVENT BOX
gtk_container_add (GTK_CONTAINER (event_box), image);

/// COLLEGO L'EVENTO ALLA FUNZIONE "helloWorld"
g_signal_connect (G_OBJECT (event_box), "button_press_event", G_CALLBACK (helloWorld), NULL);
```

***NB**: La funzione che viene collegata tramite la g_signal_connect (nell'esempio la funzione "helloWord") ha nel caso della event box un parametro in più di tipo **GdkEventButton *** .*

*Esempio:* `void helloword(GtkWidget* , GdkEventButton*);`