Programmation d'interfaces

Cours 2 - Les composants de base de *GTK*

H. Djerroud

LIASD - Université Paris 8

Automne 2020

Objectif

- La librairie *GTK* est composée d'un ensemble de composants graphiques appelés *Widgets* qui sont organisés hiérarchiquement, le but de ce cours est de vous initier à l'utilisation de ces composants.

Plan de cours

- La boucle principale et son initialisation à l'aide de *GTK*
- Compiler un programme (*GTK*)
- Les Widgets et leurs organisation hiérarchique (*GtkWidget*)
- Les Fenêtres (*GtkWindow*)
- Les Labels (*GtkLabel*)
- Les Boutons (*GtkButton*)
- Les Conteneurs (*GtkBox*)
- Les Grilles (*GtkGrid*)
- Les champs de saisie (*GtkEntry*)
- Les décorations (*GtkFrame* et *GtkSeparator*)
- Les cadres (*GtkFrame*)
- Les images (*GtkImage*)

Premier programme avec GTK

```
/* exemple1.c */
#include <gtk/gtk.h>
int main (int argc, char *argv[]){
    GtkWidget *window;
    gtk_init (&argc, &argv);
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_widget_show (window);
    gtk_main ();
    return 0;
}
```

Compiler avec GTK

gcc 'pkg-config –cflags –libs gtk+-3.0' exemple1.c -o exemple1

- Utiliser pkg-config pour définir les paramètres de compilation
- c'est mieux que d'écrire :

gcc -pthread -I/usr/include/gtk-3.0 -I/usr/include/at-spi2-atk/2.0
-I/usr/include/at-spi-2.0 -pthread -I/usr/include/gtk-3.0
-I/usr/include/at-spi2-atk/2.0 -I/usr/include/at-spi-2.0 -I/usr/include/dbus-1.0
-I/usr/lib/x86_64-linux-gnu/dbus-1.0/include -I/usr/include/gtk-3.0
-I/usr/include/gio-unix-2.0 -I/usr/include/cairo -I/usr/include/libdrm
-I/usr/include/pango-1.0 -I/usr/include/harfbuzz -I/usr/include/pango-1.0
-I/usr/include/fribidi -I/usr/include/atk-1.0 -I/usr/include/cairo
-I/usr/include/pixman-1 -I/usr/include/freetype2 -I/usr/include/libpng16
-I/usr/include/gdk-pixbuf-2.0 -I/usr/include/libmount -I/usr/include/blkid
-I/usr/include/uuid -I/usr/include/glib-2.0
-I/usr/lib/x86_64-linux-gnu/glib-2.0/include -lgtk-3 -lgdk-3 -lpangocairo-1.0
-lpango-1.0 -latk-1.0 -lcairo-gobject -lcairo -lgdk_pixbuf-2.0 -lgio-2.0
-lgobject-2.0 -lglib-2.0 exemple1.c -o exemple1

Premier programme avec GTK : application

```
#include <gtk/gtk.h>
void main_app(GtkApplication* app, gpointer user_data){
  GtkWidget *window = gtk_application_window_new (app);
  gtk_window_set_title (GTK_WINDOW (window), "Exemple");
  gtk_window_set_default_size (GTK_WINDOW (window), 200, 200);
  gtk_widget_show_all (window);
}
int main (int argc, char* argv[]){
  int ret;
  GtkApplication* app;
  app = gtk_application_new("info.halim.exo",G_APPLICATION_FLAGS_NONE);
  g_signal_connect(app,"activate", G_CALLBACK(main_app),NULL);
  ret =  g_application_run(G_APPLICATION(app),argc, argv);
  g_object_unref(app);
  return ret;
}
```

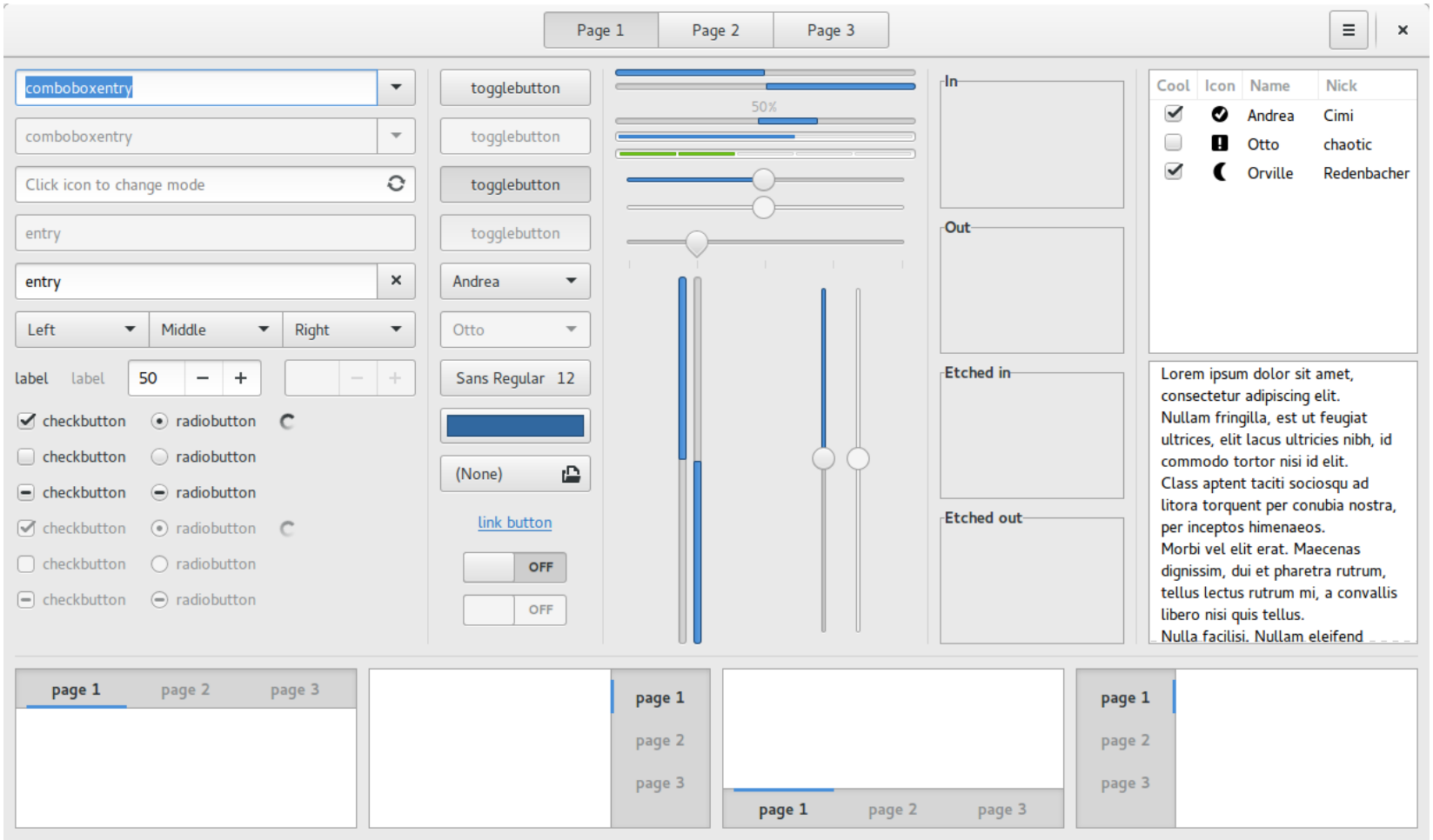
Initialisation

- `gtk_init (argc, argv);` Ajoute les paramères suivants :

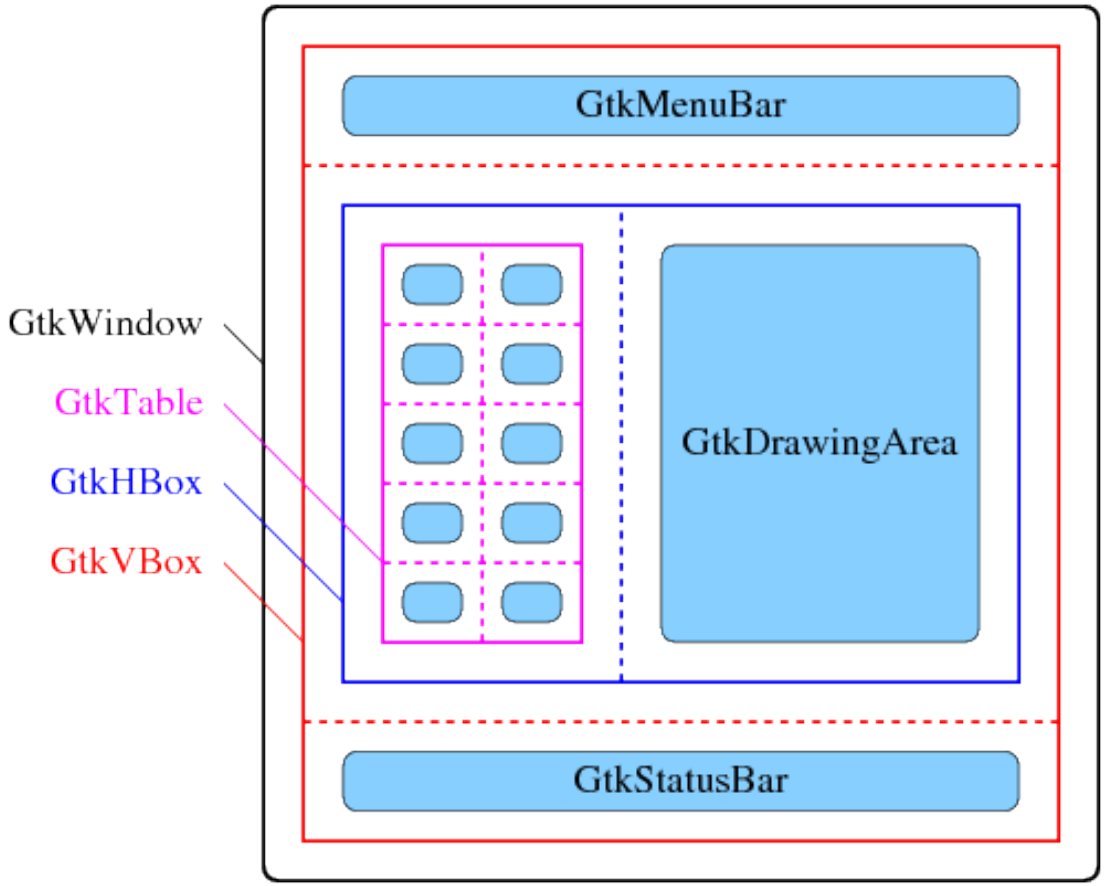
```
Utilisation :
  a.out [OPTION...]
Options de l'aide :
  -h, --help                  Affiche les options de l'aide
  --help-all                  Affiche toutes les options de l'aide
  --help-gapplication         Afficher les options GApplication
  --help-gtk                  Affiche les options GTK+

Options GApplication
  --gapplication-service      Entrer dans le mode de service GApplication (utiliser à
                              partir des fichiers de service D-Bus)
Options GTK+
  --class=CLASSE              Classe du programme telle qu'utilisée par le gestionnaire de fenêtres
  --name=NOM                  Nom du programme tel qu'utilisé par le gestionnaire de fenêtres
  --gdk-debug=DRAPEAUX        Drapeaux de débogage GDK à définir
  --gdk-no-debug=DRAPEAUX     Drapeaux de débogage GDK à ne pas définir
  --gtk-module=MODULES        Charge des modules GTK+ additionnels
  --g-fatal-warnings          Rend tous les avertissements fatals
  --gtk-debug=DRAPEAUX        Drapeaux de débogage GTK+ à définir
  --gtk-no-debug=DRAPEAUX     Drapeaux de débogage GTK+ à ne pas définir
Options de l'application :
  --display=AFFICHAGE         Affichage X à utiliser
```

Exemple GTK

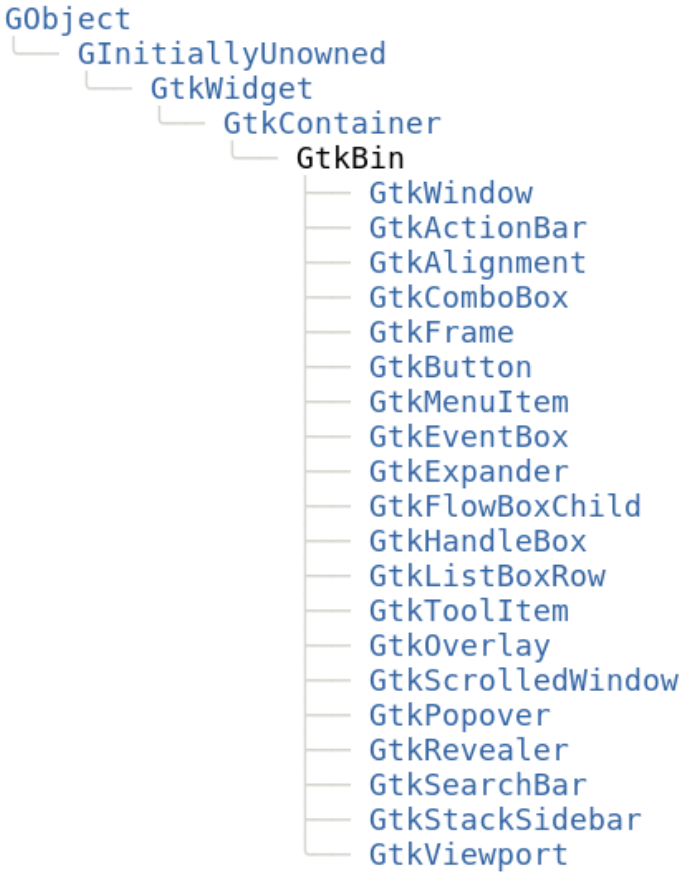

Page 1
Page 2
Page 3
comboboxentry
comboboxentry
Click icon to change mode
entry
entry
Left
Middle
Right
label
label
50
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
togglebutton
togglebutton
togglebutton
togglebutton
Andrea
Otto
Sans Regular 12
(None)
link button
OFF
OFF
50%
In
Out
Etched in
Etched out
Cool
Icon
Name
Nick
Andrea
Cimi
Otto
chaotic
Orville
Redenbacher
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam fringilla, est ut feugiat ultrices, elit lacus ultricies nibh, id commodo tortor nisi id elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Morbi vel elit erat. Maecenas dignissim, dui et pharetra rutrum, tellus lectus rutrum mi, a convallis libero nisi quis tellus. Nulla facilisi. Nullam eleifend
page 1
page 2
page 3
page 1
page 2
page 3
page 1
page 2
page 3
page 1
page 2
page 3

Organisation d'une fenêtre



Les fenêtres

- Une GtkWindow est une fenêtre de premier niveau qui peut contenir d'autres widgets. Les fenêtres ont normalement des décorations qui sont sous le contrôle du WM (système de fenêtrage) et permettent à l'utilisateur de manipuler la fenêtre (la redimensionner, la déplacer, la fermer, ...).



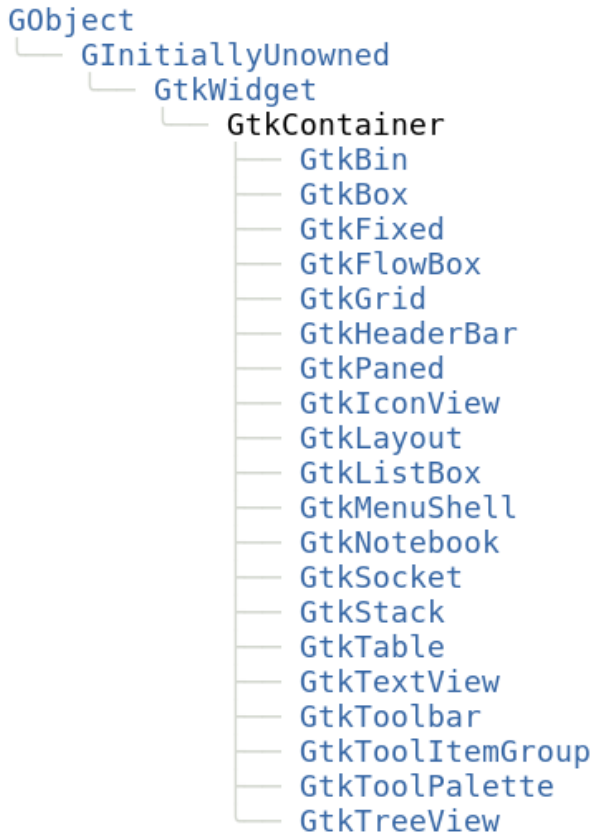
Exemple création d'une fenêtre

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

int main(int argc, char* argv[]){
  GtkWidget* win;
  gtk_init(&argc, &argv);
  win  = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_widget_show_all(win);
  gtk_main();
  return 0;
}
```

Les conteneurs

- Les widgets de conteneur sont les nœuds internes de l'arborescence des widgets résultante : ils contiennent d'autres widgets. Ainsi, par exemple, vous pourriez avoir une GtkWindow contenant un GtkFrame contenant un GtkLabel. Si vous vouliez une image au lieu d'une étiquette textuelle à l'intérieur du cadre, vous pouvez remplacer le widget GtkLabel par un widget GtkImage.



Les conteneurs

Il existe deux principaux types de widgets de conteneur :

- widget conteneur a un widget enfant unique et dérive de GtkBin. Ces conteneurs sont des décorateurs, qui ajoutent une sorte de fonctionnalité à l'enfant. Par exemple, un GtkButton transforme son enfant en un bouton cliquable ; un GtkFrame dessine un cadre autour de son enfant et un GtkWindow place son widget enfant dans une fenêtre de niveau supérieur.
- Le deuxième type de conteneur peut avoir plus d'un enfant ; son objectif est de gérer la mise en page. Cela signifie que ces conteneurs attribuent des tailles et des positions à leurs enfants. Par exemple, un GtkHBox organise ses enfants dans une ligne horizontale et un GtkGrid organise les widgets qu'il contient dans une grille bidimensionnelle.

Exemple 1 conteneur à enfant unique

```
#include <gtk/gtk.h>
int main(int argc, char* argv[]){
  GtkWidget* win, *btn, *lab;
  gtk_init(&argc, &argv);
  win = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  btn = gtk_button_new();
  lab = gtk_label_new("Bouton exemple");
  gtk_container_add (GTK_CONTAINER(btn),lab);
  gtk_container_add (GTK_CONTAINER(win),btn);
  gtk_widget_show_all(win);
  gtk_main();
  return 0;
}
```

Exemple 2 conteneur à enfants multiples

```
#include <gtk/gtk.h>
int main(int argc, char* argv[]){
  GtkWidget* win, *bt1, *bt2, *hbox;
  gtk_init(&argc, &argv);
  win  = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  hbox = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 20);
  bt1  = gtk_button_new_with_label ("Bouton 1 ");
  bt2  = gtk_button_new_with_label ("Bouton 2 ");
  gtk_container_add(GTK_CONTAINER(win), hbox);
  gtk_box_pack_start (GTK_BOX(hbox), bt1, TRUE, TRUE, 0);
  gtk_box_pack_start (GTK_BOX(hbox), bt2, TRUE, TRUE, 0);
  gtk_widget_show_all(win);
  gtk_main();
  return 0;
}
```

Les conteneurs : Sur une ligne

- Le widget GtkBox organise les widgets enfants en une seule ligne ou colonne, selon la valeur de sa propriété ≪orientation≫. Dans l'autre dimension, tous les enfants ont la même taille.

```
GtkWidget * gtk_box_new (GtkOrientation ori, gint spac);
```

Les valeurs possible pour l'orientation :

- GTK_ORIENTATION_HORIZONTAL
- GTK_ORIENTATION_VERTICAL

Les conteneurs : Sur une grille

- GtkGrid est un conteneur qui organise ses widgets enfants en lignes et en colonnes, avec des positions arbitraires et des étendues horizontales / verticales.

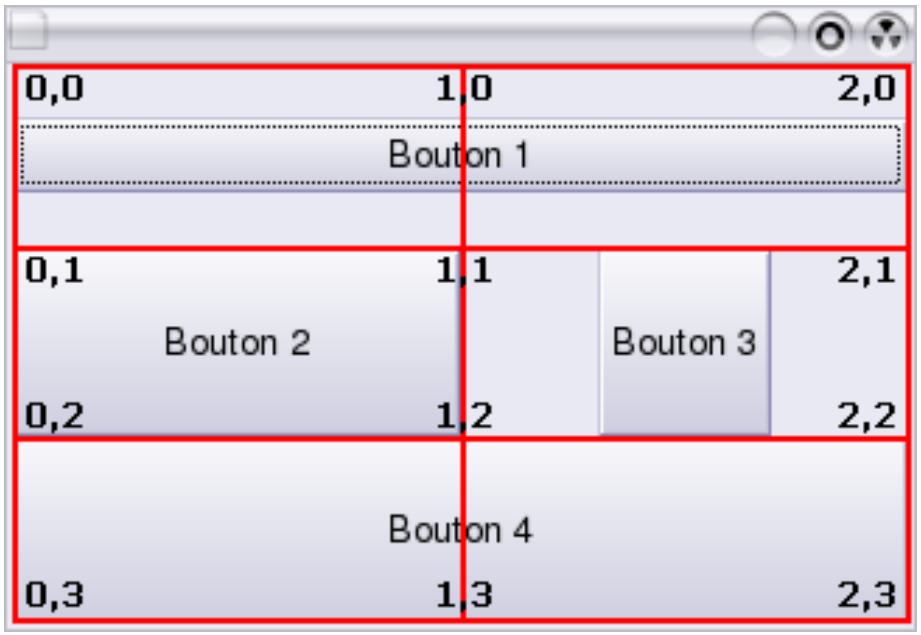
```
GtkWidget * gtk_grid_new (void);
```

Pour ajouter les éléments : `void gtk_grid_attach (GtkGrid *grid, GtkWidget *child, gint left, gint top, gint width, gint height);`

- `grid` : un GtkGrid
- `child` : le widget à ajouter
- `left` : le numéro de colonne auquel attacher le côté gauche de l'enfant
- `top` : le numéro de ligne auquel attacher la partie supérieure de l'enfant
- `width` : le nombre de colonnes que l'enfant couvrira
- `width` : le nombre de lignes que l'enfant couvrira

Les conteneurs : Sur une grille

- La disposition se fait comme suite :



Les boutons

```
GtkWidget* gtk_button_new(void);
```

- Créer un bouton vide

```
GtkWidget* gtk_button_new_with_label(const gchar *label);
```

- Créer un bouton avec un label à l'intérieur du bouton

```
GtkWidget* gtk_button_new_with_mnemonic(const gchar *label);
```

- Créer un bouton avec un label, et le faire réagir le bouton à l'aide d'un raccourci clavier

```
GtkWidget* gtk_button_new_from_stock(const gchar *stock_id);
```

- Créer un bouton avec un label, un raccourci clavier et une image

Les champs de saisie

```
GtkWidget* gtk_entry_new(void);
```

- Crée un widget GtkEntry de base

```
GtkWidget* gtk_entry_new_with_max_length(const gchar *label);
```

- Limite de nombre maximum de caractère que l'utilisateur peut saisir