INTRODUCTION
A
GTK+ 2

UNE LIBRAIRIE GRAPHIQUE
MULTI-PLATEFORME - MULTI-LANGAGE

INTRODUCTION A GTK+ 2

UNE LIBRAIRIE GRAPHIQUE MULTI-PLATEFORME MULTI-LANGAGE

INTRODUCTION

GTK+ est une librairie permettant de créer des interfaces graphiques conçut à l'origine pour le projet GIMP et aujourd'hui étendu à tout type d'application. L'avantage de cette librairie est d'une part sa gratuité (diffusée en open source sous licence LGPL), et d'autre part son utilisation multi langage (C, C++, Perl, Python, etc.) et multi plateforme (Windows, Linux, BSD, Beos, etc.). Pour plus d'informations, le site officiel est disponible à l'adresse : http://www.gtk.org

INSTALLATION

L'installation de GTK+ sous Windows avec l'IDE DevCpp est assez aisée.

Il vous faut télécharger et décompresser le Pack Windows disponible sur le site officiel sur votre disque dur (ex. dans c:\Dev-Cpp\GTK\).

Puis définir la variable d'environnement du chemin du dossier soit sur Windows 2K/XP : Panneau de configuration > Système > Onglet « Avancé » > Variables d'environnement et ajouter le chemin du dossier « bin » à la variable système PATH (ex : c:\Dev-Cpp\GTK\GTK-2.4.9\bin).

Ensuite, copier les deux fichiers (« gtk.template » et « gtk_c.txt ») contenu dans le dossier « Dev-Cpp » du Pack Windows dans le dossier « Template » de l'IDE (ex : c:\Dev-Cpp\Template).

Enfin, il vous faudra éditer le fichier « Gtk.template » précédemment copié pour modifier les variables « Compiler » et « Linker » (lignes 17 et 18) en remplaçant la chaîne « C:\GTK-2.4.9 » par le chemin de votre dossier GTK (dans l'exemple : c:\Dev-Cpp\GTK\GTP-2.4.9).

Pour les autres IDE et/ou autres OS, consulter le fichier README fourni avec GTK ou consulter le site officiel.

PREMIER PROGRAMME GTK+

Lancer Dev-Cpp, et créer un nouveau projet C nommé « GTKDemo » en sélectionnant « GTK+ » dans l'onglet « GUI » comme type de projet.

Un code de base (template) est automatiquement généré. Sans rien modifier, compiler et exécuter le code.

Le programme devrait lancer une fenêtre Windows vide ! GTK est donc opérationnel et nous pouvons commencer l'écriture d'application graphique.

Auteur : sebastien.warin@supinfo.com

NOTIONS DE BASE

La création d'interface graphique sous GTK+ consiste à créer et manipuler des objets graphiques de GTK+. Ces objets sont appelés « Widgets ». Un Widget est une structure proposant des fonctions et propriétés permettant la manipulation de ces objets.

Bien que nous sommes dans un langage procédural, le terme « objet » est à prendre au sens littéral car GTK+ introduit la notion d'héritage. Les objets graphiques héritent des membres d'un widget de base : GtkWidget.

La gestion des événements est effectuée dans une boucle événementielle par l'interception de « signaux ». Un click sur un bouton par exemple, déclenchera un signal intercepté par la boucle événementielle qui exécutera la fonction correspondante. On appelle « fonction callback » une fonction associée à un signal.

LES FENETRES

Le widget GtkWindow est l'élément de base d'une interface graphique.

Analysons le code de base de la Template GTK+ :

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

int main(int argc, char **argv)
{
        GtkWidget *pWindow;
        gtk_init(&argc, &argv);
        pWindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        g_signal_connect(G_OBJECT(pWindow),"destroy",
G_CALLBACK(gtk_main_quit), NULL);
        gtk_widget_show_all(pWindow);
        gtk_main();
        return EXIT_SUCCESS;
}
```

Tout d'abord nous remarquons l'inclusion du fichier d'entête « gtk.h » indispensable pour la création de programme utilisant GTK+.

La première ligne de la fonction main() crée un pointeur nommé pWindow sur un type GtkWidget qui sera notre widget de base (fenêtre).

La fonction gtk_init() est appelée dans toutes les applications GTK+, elle initialise la bibliothèque et configure certains paramètres comme l'aspect visuel, les couleurs, etc.

La fonction gtk_window_new() initialise notre objet pWindow en renvoyant le pointeur sur l'objet. La constante GTK_WINDOW_TOPLEVEL sert à créer une fenêtre complète visible dans la barre des taches.

La ligne suivante appelle la fonction g_signal_connect() qui permet de connecter un événement (signal) à un objet. Ici, il y a connexion de l'événement « destroy » sur notre objet pWindow. Cet événement sera déclenché à la

fermeture de la fenêtre et appellera la fonction callback gtk_main_quit() qui fermera la boucle événementielle.

L'instruction suivante, gtk_widget_show_all() est une procédure permettant d'afficher le widget passé en paramètre ainsi que tous les widgets qu'il contient. Ici elle vient afficher notre fenêtre pWindow.

Pour finir le programme appelle la fonction gtk_main() permettant de démarrer la boucle événementielle.

On pourrait compléter un peu plus ce code pour y définir un titre à notre fenêtre, une taille, sa position par les lignes :

```
/* Définition de la position au centre */
gtk_window_set_position(GTK_WINDOW(pWindow), GTK_WIN_POS_CENTER);
/* Définition de la taille de la fenêtre 320x200 */
gtk_window_set_default_size(GTK_WINDOW(pWindow), 320, 200);
/* Titre de la fenêtre : GTK-Demo */
gtk_window_set_title(GTK_WINDOW(pWindow), "GTK-Demo by SeBeuH");
```

WIDGETS DE BASE : LABEL, BOUTONS, BOX ET TEXTBOX

LES LABELS

C'est le widget GtkLabel qui nous permet d'afficher du texte dans une fenêtre.

La création d'un label se fait grâce à la fonction gtk_label_new() qui prend en paramètre le texte du label et renvoi un pointeur sur le widget.

Il faut ensuite ajouter notre widget label dans notre fenêtre par la fonction gtk_container_add() prenant en paramètre le conteneur soit notre widget pWindow et le widget label précédemment créé.

Rajoutez le code ci-dessus dans la template de base afin de visualiser le résultat :

```
/* Déclaration du pointeur pour le label */
GtkWidget *pLabel;
/* Création du label */
pLabel=gtk_label_new("Demo GTK+ by SeBeuH");
/* Ajout du label a l'intérieur de la fenêtre */
gtk_container_add(GTK_CONTAINER(pWindow), pLabel);
```

Nous pouvons aussi utiliser la fonction gtk_label_set_label() pour redéfinir le texte du label. La fonction prend en paramètre le pointeur du widget du label et le texte.

Pour récupérer le texte du label, on utilise la fonction gtk_label_get_label() qui prend en paramètre le pointeur du widget label et renvoie une constante de type gchar.

LES BOUTONS

Les boutons sont disponibles grâce au widget GtkButton. Il y a quatre manières d'initialiser notre bouton :

- Gtk_button_new : crée un bouton vide permettant ensuite de le personnaliser complètement en le remplissant d'un autre widget comme un label, images, etc…

- Gtk_button_new_with_label : crée un bouton contenant un label dont le texte sera définit en paramètre.

- Gtk_button_new_with_mnemonic : identique à la précédente mais qui permet en plus de définir une touche de raccourci clavier.

- Gtk_button_new_from_stock : crée un bouton avec un label, un raccourci et une image. Toutes ces informations sont contenues dans un objet GtkStockItem. La fonction prend donc en paramètre le pointeur sur un GtkStockItem définit préalable avant.

Une fois le bouton créé, il nous reste plus qu'à le connecter à un signal pour au moins le faire réagir au click puis de l'injecter dans une fenêtre.

Dans notre exemple, nous allons rajouter un bouton pour quitter l'application :

```
/* Déclaration du pointeur pour le Button */
GtkWidget *pButton;
/* Création du bouton avec un label */
pButton = gtk_button_new_with_label("Quitter");
/* Connexion du signal "clicked" du bouton */
g_signal_connect(G_OBJECT(pButton), "clicked",
G_CALLBACK(gtk_main_quit), NULL);
/* Insertion du bouton dans la fenêtre */
gtk_container_add(GTK_CONTAINER(pWindow), pButton);
```

Pour la modification et récupération du texte du bouton, c'est le même principe que pour les label avec les fonctions: gtk_button_get_label() et gtk_button_set_label().

LES BOX

Si vous avez rajouté le code ci-dessus vous êtes probablement tombé sur une erreur car le widget GtkContainer ne peut contenir qu'un seul widget. C'est pourquoi on utilise le widget GtkBox permettant lui, d'en inclure plusieurs.

Il existe de type de GtkBox :

- GtkHBox : dispose les widgets horizontalement

- GtkVBox : idem mais verticalement

Les fonctions de création sont respectivement gtk_hbox_new() et gtk_vbox_new() prenant deux arguments : un gboolean (true ou false) nommé « homogeneous » qui définit si les widgets qu'il contient utilisent un espace équivalent et un gint pour définir l'espacement à l'écran des widgets.

Les fonctions d'ajout sont gtk_box_pack_start() et gtk_box_pack_end(). On rajoute les widgets dans un ordre, soit au début soit à la fin. Ces fonctions prennent cinq paramètres en entrée :

- Un pointeur sur un GtkBox : indique la Box a remplir.
- Un pointeur sur un GtkWidget : indique le Widget a injecter.
- Un gBoolean « expand » : indique si l'on partage ou non l'espace restant dans la box avec les autres widgets.
- Un gBoolean « fill » : indique si le widget enfant occupe toute la zone qui lui est réservée.
- Un gUInt « padding » : définit les marges autour du widget.

Reprenons donc les exemples précédents pour créer une application contenant un label et un bouton. Le click du bouton modifiera le texte du label ce qui nous donne :

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

void change_label (GtkWidget *widget, gpointer data)
{
    gtk_label_set_label(data,"Le GTK+ c'est cool :p");
}

int main(int argc, char **argv)
{
    /* Déclaration des pointeurs pour les Widgets */
    GtkWidget *pWindow;
    GtkWidget *pLabel;
    GtkWidget *pButton;
    GtkWidget *pVBox;

    /* Initialisation de GTK+*/
    gtk_init(&argc, &argv);

    /* Création de la fenêtre */
    pWindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    /* Définition de la position au centre */
    gtk_window_set_position(GTK_WINDOW(pWindow), GTK_WIN_POS_CENTER);
    /* Définition de la taille de la fenêtre 320x100 */
    gtk_window_set_default_size(GTK_WINDOW(pWindow), 320, 100);
    /* Titre de la fenêtre : GTK-Demo */
    gtk_window_set_title(GTK_WINDOW(pWindow), "GTK-Demo by SeBeuH");

    /* Création de la GtkBox verticale */
    pVBox = gtk_vbox_new(TRUE, 0);

    /* Création du label */
    pLabel=gtk_label_new("Bienvenue !");
    /* Création du bouton */
    pButton = gtk_button_new_with_label("Test");

    /* Connexion du signal "destroy" de la fenêtre */
    g_signal_connect(G_OBJECT(pWindow), "destroy",
G_CALLBACK(gtk_main_quit), NULL);
    /* Connexion du signal "clicked" du bouton */
    g_signal_connect(G_OBJECT(pButton), "clicked",
G_CALLBACK(change_label), pLabel);
```

```
    /* Injection du Label et Button dans la VBox*/
    gtk_box_pack_start(GTK_BOX(pVBox), pLabel, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pButton, TRUE, FALSE, 0);

    /* Ajout de la GtkVBox dans la fenêtre */
    gtk_container_add(GTK_CONTAINER(pWindow), pVBox);

    /* Affichage et démarrage de la boucle even.*/
    gtk_widget_show_all(pWindow);
    gtk_main();

    /* Fin ! */
    return EXIT_SUCCESS;
}
```

Rappelons que les événements sont définit par la fonction g_signal_connect(). Cette fonction prend quatre arguments en entrée. Le premier est le widget où sera appliqué l'événement, le second est le type de l'événement, le troisième est la fonction callback à appeler lors du déclenchement de l'événement et enfin le dernier définit le pointeur de données.

L'événement « detroy » de l'objet pWindow appelle la fonction gtk_main_quit() pour quitter l'application et le pointeur de donnée est défini a NULL.

Par contre l'événement « clicked » de l'objet pButton appelle la fonction change_label() et passe dans le paramètre « pointeur de données » le pointeur sur l'objet pLabel. La fonction change_label() appelle la fonction gtk_label_set() en passant en paramètre l'argument « data » qui contient le pointeur du label. Ainsi notre pLabel sera modifié.

Enfin pour finir il existe une autre famille de containers plus intéressante que les Box : ce sont les Tables (GtkTables) qui utilisent une grille invisible pour placer nos widgets. (cf. le manuel de référence plus d'informations.)

LES TEXTBOX

Les Textbox ou zone de saisie sont disponibles sous le widget GtkEntry. Ils sont créés par la fonction gtk_entry_new().

La récupération et modification se font respectivement par les fonctions gtk_entry_get_text() et gtk_entry_set_text(). Nous pouvons aussi définir le nombre de caractères maximum par la fonction gtk_entry_set_max_length() qui prend en paramètre le pointeur du widget GtkEntry et un gInt définissant le nombre maximum de caractères.

Modifions notre programme précèdent en rajoutant une textbox. Le click sur le bouton affichera le texte saisi dans la texbox dans notre label.

Pour cela rajoutons dans les définitions notre Widget GtkEntry nommé pEntry par la ligne :

```
GtkWidget *pEntry;
```

Rajoutons également la création de notre widget par la ligne :

```
pEntry = gtk_entry_new();
```

Et enfin rajoutons notre widget dans notre box :

```
gtk_box_pack_start(GTK_BOX(pVBox), pEntry, TRUE, FALSE, 0);
```

Il nous faut maintenant passer un second argument dans la fonction callback change_label() pour « envoyer » le pointeur de la texbox (pEntry).

Comme nous pouvons passer en paramètre qu'un seul pointeur (gpointer), nous allons utiliser un GPtrArray qui est un tableau de pointeur. Rajoutons donc avant la connexion du signal « clicked » de notre objet pButton le code :

```
GPtrArray *args;
args = g_ptr_array_new();
g_ptr_array_add(args, pLabel);
g_ptr_array_add(args, pEntry);
```

Ces lignes code vont créer un pointeur sur un tableau de pointeur nommé « args ». Ce tableau de pointeur est créé par la fonction g_ptr_array_new(). La fonction g_ptr_array_add() permet de remplir notre tableau de pointeur. Le premier argument étant le tableau de pointeur a remplir et le second étant le pointeur a insérer dans notre tableau. Ici nous remplissons notre tableau « args » du pointeur sur pLabel et pEntry.

Il faut bien sûr modifier la connexion du signal « clicked » de notre pButton en passant comme paramètres pour « pointeur de données » le tableau « args » :

```
g_signal_connect(G_OBJECT(pButton), "clicked",
G_CALLBACK(change_label), args);
```

Pour finir, il faut modifier notre fonction change_label() par le code :

```
GPtrArray *array = (GPtrArray *)data;
gtk_label_set_label(array->pdata[0],gtk_entry_get_text(array-
>pdata[1]));
```

La première ligne crée un GPtrArray nommé « array » qui récupère le tableau de pointeur passé en paramètre (data).

La seconde ligne vient modifier le label de « array->pdata[0] », qui est le premier pointeur de notre tableau qui correspond à notre objet pLabel, par le texte de « gtk_entry_get_text(array->pdata[1]) », qui récupère le texte du widget « array->pdata[1] » qui correspond au deuxième pointeur de notre tableau soit pEntry.

Le code final nous donne donc :

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

void change_label (GtkWidget *widget, gpointer data)
{
    /* Recuperation du tableau de pointeur */
    GPtrArray *array = (GPtrArray *)data;
    /* Modification du texte de pLabel par le texte de pEntry */
    gtk_label_set_label(array->pdata[0],
gtk_entry_get_text(array->pdata[1]));
}
```

```
int main(int argc, char **argv)
{
    /* Déclaration des pointeurs pour les Widgets */
    GtkWidget *pWindow;
    GtkWidget *pLabel;
    GtkWidget *pButton;
    GtkWidget *pVBox;
    GtkWidget *pEntry;
    /* Déclaration du tableau de pointeur */
    GPtrArray *args;

    /* Initialisation de GTK+*/
    gtk_init(&argc, &argv);

    /* Création de la fenêtre */
    pWindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    /* Définition de la position au centre */
    gtk_window_set_position(GTK_WINDOW(pWindow),
GTK_WIN_POS_CENTER);
    /* Définition de la taille de la fenetre 320x100 */
    gtk_window_set_default_size(GTK_WINDOW(pWindow), 320, 100);
    /* Titre de la fenêtre : GTK-Demo */
    gtk_window_set_title(GTK_WINDOW(pWindow), "GTK-Demo by
SeBeuH");

    /* Création de la GtkBox verticale */
    pVBox = gtk_vbox_new(TRUE, 0);

    /* Création du label */
    pLabel=gtk_label_new("Bienvenue !");
    /* Création du bouton */
    pButton = gtk_button_new_with_label("Test");
    /* Création du textbox */
    pEntry = gtk_entry_new();
    /* Création du tableau de pointeur*/
    args = g_ptr_array_new();

    /* Remplissage du tableau de pointeurs de pLabel et pEntry
*/
    g_ptr_array_add(args, pLabel);
    g_ptr_array_add(args, pEntry);

    /* Connexion du signal "destroy" de la fenetre */
    g_signal_connect(G_OBJECT(pWindow), "destroy",
G_CALLBACK(gtk_main_quit), NULL);
    /* Connexion du signal "clicked" du bouton */
    g_signal_connect(G_OBJECT(pButton), "clicked",
G_CALLBACK(change_label), args);

    /* Injection du Label et Button dans la VBox*/
    gtk_box_pack_start(GTK_BOX(pVBox), pLabel, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pButton, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pEntry, TRUE, FALSE, 0);

    /* Ajout de la GtkVBox dans la fenêtre */
    gtk_container_add(GTK_CONTAINER(pWindow), pVBox);

    /* Affichage et démarrage de la boucle even.*/
    gtk_widget_show_all(pWindow);
    gtk_main();
```

```
        /* Fin ! */
        return EXIT_SUCCESS;
    }
```

WIDGETS AVANCES

GTK+ offre des widgets plus avancés comme :

- Les tables
- Les listes chaînées
- Les décorations
- Les images
- Les boite de dialogues
- Les menus
- Les barres d'outils
- Les barres d'état
- Les barres de progression
- La sélection de fichier
- Les listes et arbres
- Etc. (liste complète dans le manuel de référence)

Cette documentation est une introduction à GTK+, c'est pourquoi nous ne détaillerons pas tout ces widgets car il serait bien trop long de tout passer en revue. Pour plus d'information sur leur utilisation je vous renvoi donc vers le manuel de référence (cf. Sources d'informations a la fin de ce document).

Dans le dernier exemple, nous allons retravailler notre programme précèdent pour lui rajouter une barre de status (StatusBar) qui affichera un simple texte, une barre de menu et ainsi que deux boites de dialogue : une pour la confirmation de la fermeture du programme (« Fichier > Quitter ») et une pour afficher un simple message pour le click sur « ? > A propos de … ».

Le code :

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

void change_label (GtkWidget *widget, gpointer data)
{
    /* Recuperation du tableau de pointeur */
    GPtrArray *array = (GPtrArray *)data;
    /* Modification du texte de pLabel par le texte de pEntry */
    gtk_label_set_label(array->pdata[0],gtk_entry_get_text(array-
>pdata[1]));
}
```

```
void Quitter(GtkWidget* widget, gpointer data)
{
    /* Déclaration du Widget */
    GtkWidget *pQuestion;

    /* Création de la boite de dialogue de type question (Yes/No) */
    pQuestion = gtk_message_dialog_new(GTK_WINDOW(data),
        GTK_DIALOG_MODAL,
        GTK_MESSAGE_QUESTION,
        GTK_BUTTONS_YES_NO,
        "Voulez vous vraiment\n"
        "quitter le programme?");
    /* Affichage de pQuestion et Action en fonction du résultat de
retour */
    switch(gtk_dialog_run(GTK_DIALOG(pQuestion)))
    {
        case GTK_RESPONSE_YES:
             // Oui -> Quitter le programme (gtk_main_quit())
            gtk_main_quit();
            break;
        case GTK_RESPONSE_NONE:
        case GTK_RESPONSE_NO:
             // Rien ou Non --> Destruction du widget (pQuestion)
            gtk_widget_destroy(pQuestion);
            break;
    }
}

void About(GtkWidget* widget, gpointer data)
{
    /* Déclaration du Widget */
    GtkWidget *pAbout;
    /* Création de la boite de dialogue de type information */
    pAbout = gtk_message_dialog_new (GTK_WINDOW(data),
        GTK_DIALOG_MODAL,
        GTK_MESSAGE_INFO,
        GTK_BUTTONS_OK,
        "Introduction a GTK+ 2\n"
        "Par WARIN Sebastien (SeBeuH)");
     /* Affichage de pAbout */
     gtk_dialog_run(GTK_DIALOG(pAbout));
     /* Destruction du widget (pAbout) */
     gtk_widget_destroy(pAbout);
}

int main(int argc, char **argv)
{
      /* Declaration des Widgets*/
      GtkWidget *pWindow;
      GtkWidget *pLabel;
      GtkWidget *pButton;
      GtkWidget *pVBox;
      GtkWidget *pEntry;
      GtkWidget *pMenuBar;
      GtkWidget *pMenu;
      GtkWidget *pMenuItem;
      GtkWidget *pStatusBar;
      /* Déclaration du tableau de pointeur */
      GPtrArray *args;
```

```
    /* Initialisation de GTK+*/
    gtk_init(&argc, &argv);

    /* Création de la fenêtre */
    pWindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);


    /* Définition de la position au centre */
    gtk_window_set_position(GTK_WINDOW(pWindow), GTK_WIN_POS_CENTER);
    /* Définition de la taille de la fenêtre 320x100 */
    gtk_window_set_default_size(GTK_WINDOW(pWindow), 320, 100);
    /* Titre de la fenêtre : GTK-Demo */
    gtk_window_set_title(GTK_WINDOW(pWindow), "GTK-Demo by SeBeuH");

    /* Création de la GtkBox verticale */
    pVBox = gtk_vbox_new(TRUE, 0);

    /* Création du label */
    pLabel=gtk_label_new("Bienvenue !");
    /* Création du bouton */
    pButton = gtk_button_new_with_label("Test");
    /* Création du textbox */
    pEntry = gtk_entry_new();
    /* Création de la StatusBar */
    pStatusBar = gtk_statusbar_new();
    /* Ajout du texte dans la StatusBar */
    gtk_statusbar_push (GTK_STATUSBAR (pStatusBar), 0, "Demo GTK 2+
by SebeuH");

    /* Création de la barre de menu */
    pMenuBar = gtk_menu_bar_new();

    /*
       Création du 1er sous-menu :
    */
       // Création du menu
    pMenu = gtk_menu_new();
       // Nouveau (création du pMenuItem & ajout dans pMenu) :
    pMenuItem = gtk_menu_item_new_with_label("Nouveau");
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
       // Ouvrir :
    pMenuItem = gtk_menu_item_new_with_label("Ouvrir");
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
       // Enregistrer :
    pMenuItem = gtk_menu_item_new_with_label("Enregistrer");
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
       // Fermer:
    pMenuItem = gtk_menu_item_new_with_label("Fermer");
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
       // Quitter (+ connexion au signal quand activé sur la fct
Quitter())
    pMenuItem = gtk_menu_item_new_with_label("Quitter");
    g_signal_connect(G_OBJECT(pMenuItem), "activate",
G_CALLBACK(Quitter), (GtkWidget*) pWindow);
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
    /* Création du Menu Fichier  */
    pMenuItem = gtk_menu_item_new_with_label("Fichier");
    /* Ajout dans Fichier du sous-menu contenu dans pMenu */
    gtk_menu_item_set_submenu(GTK_MENU_ITEM(pMenuItem), pMenu);
    /* Ajout du menu Fichier dans la barre de menu (pMenuBar) */
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenuBar), pMenuItem);
```

```
    /*
       Création du 2eme sous-menu :
    */
       // Création du menu :
    pMenu = gtk_menu_new();
       // A propos (+ connexion du signal activate sur la fct
About()) :
    pMenuItem = gtk_menu_item_new_with_label("A propos de...");
    g_signal_connect(G_OBJECT(pMenuItem), "activate",
G_CALLBACK(About), (GtkWidget*) pWindow);
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenu), pMenuItem);
    /* Création du Menu ?  */
    pMenuItem = gtk_menu_item_new_with_label("?");
    /* Ajout dans ? du sous-menu contenu dans pMenu */
    gtk_menu_item_set_submenu(GTK_MENU_ITEM(pMenuItem), pMenu);
    /* Ajout du menu ? dans la brre de menu (pMenuBar) */
    gtk_menu_shell_append(GTK_MENU_SHELL(pMenuBar), pMenuItem);

    /* Création du tableau de pointeur*/
    args = g_ptr_array_new();

    /* Remplissage du tableau par pLabel et pEntry*/
    g_ptr_array_add(args, pLabel);
    g_ptr_array_add(args, pEntry);

    /* Connexion du signal "destroy" de la fenetre */
    g_signal_connect(G_OBJECT(pWindow), "destroy",
G_CALLBACK(gtk_main_quit), NULL);
    /* Connexion du signal "clicked" du bouton */
    g_signal_connect(G_OBJECT(pButton), "clicked",
G_CALLBACK(change_label), args);

    /* Injection des widgets dans la VBox*/
    gtk_box_pack_start(GTK_BOX(pVBox), pMenuBar, FALSE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pLabel, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pButton, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(pVBox), pEntry, TRUE, FALSE, 0);
    gtk_box_pack_end(GTK_BOX(pVBox), pStatusBar, FALSE, FALSE, 0);

    /* Ajout de la GtkVBox dans la fenetre */
    gtk_container_add(GTK_CONTAINER(pWindow), pVBox);

    /* Affichage et démarrage de la boucle even.*/
    gtk_widget_show_all(pWindow);
    gtk_main();

    /* Fin ! */
    return EXIT_SUCCESS;
}
```

SOURCES D'INFORMATIONS

- Le site officiel : http://www.gtk.org/
- Le tutorial officiel : http://www.gtk.org/tutorial/ et en version française : http://www.linux-france.org/article/devl/gtk/gtk_tut.html
- Le manuel de référence : http://developer.gnome.org/doc/API/2.0/gtk/index.html

CONCLUSION

Cette courte introduction à GTK+ nous aura fait découvrir la manière dont est structurée la librairie en découvrant des widgets de base comme les fenêtres, les labels, les textbox… Mais la librairie est bien trop importante pour tout détailler. Il a noté aussi que nous pouvons nous même développer d'autres Widgets en se basant sur ce que propose GTK+.

Si GTK+ vous a plu, je vous recommande vivement de suivre le tutorial français ou anglais qui fait un tour complet de la librairie puis de voir le manuel de référence pour obtenir tous les détails.

Il est marrant de constater que même dans le cadre d'un développement en langage C, soit langage procédurale, on arrive à se croire, avec GTK+, dans un langage orienté objet de part sa conception. On retrouve des « constructeurs », des « objets » composer de « membres » avec une gestion événementielle….

Personnellement je pense que GTK+ est une excellente librairie, puissante, portable, gratuite et assez simple à prendre en main mais qui peut s'avérer un cauchemar en nombre de ligne de code pour une interface complexe à moins de posséder un IDE permettant de concevoir graphiquement son interface. (A ce sujet je vous invite à regarder du coté de Glade sous Linux).

Auteur : sebastien.warin@supinfo.com