	Atelier Glade et GTK+	
	Matière : Projet 2P-3B Classe(s) : 2P/3B Equipe : Projet C	Unité pédagogique : Algorithmique & Programmation Année universitaire : 2020-2021

Objectifs :

- Découvrir par la pratique la programmation des interfaces graphiques
- Découvrir l'utilisation d'un constructeur d'interface
- Découvrir comment connecter les signaux d'une interface et comment accéder à ses composants
- Découvrir comment consulter et modifier la valeur d'un composant graphique

Partie 1 : Définitions et Concepts de base**1. GTK+**

GTK+ une bibliothèque C pour faire des interfaces graphiques. A l'origine, elle a été développée pour les besoins du logiciel du traitement d'image GIMP (GNU Image Manipulation Program). Mais aujourd'hui, GTK+ est utilisée dans d'autres projets comme la création d'interfaces graphiques multi plateformes (Windows, UNIX, BeOs). C'est une bibliothèque libre et sous licence GNU LGPL, elle est de plus disponible sur plusieurs langages tels que C++, Ada, Perl, PHP...

2. Glade

Glade est une application qui permet la conception des interfaces graphiques à la souris sans écrire aucune ligne de code. Il prend en charge toute la partie de gestion/génération de l'interface pour permettre au développeur de se concentrer sur le code utile (la partie métier).

Pour ce qui concerne notre projet on va utiliser la version glade2.12.2.

Installation de concepteur d'interface Glade 2.12.2

Voici les étapes à suivre afin d'installer glade2.12.2 :

1. Installation de dépendances

Tout d'abord vous devez vous assurer que vous avez une bonne connexion pour garantir l'installation complète des outils utilisés et pour ne pas avoir des problèmes lors de l'exécution de l'outil glade. La majorité des logiciels open source sous linux dépendent d'autres bibliothèques et outils. Dans notre cas, glade est un logiciel open source sous licence GPL V2 et nécessite la présence des dépendances suivantes dans votre système d'exploitation Linux :

- **libxml2-dev** : les fichiers de développement pour la bibliothèque XML GNOME
- **libgtk2.0-dev** : les fichiers de développement pour la bibliothèque GTK+
- **scrollkeeper** : C'est un système de catalogage de documents.

Pour installer ces dépendances exécuter la commande suivante :

- **sudo apt-get install libxml2-dev**
- **sudo apt-get install libgtk2.0-dev**
- **sudo apt-get update -y**
- **sudo apt-get install -y scrollkeeper**

Ensuite, installer les dépendances suivantes :

- **sudo apt-get build-dep libxml-parser-perl**
- **sudo apt-get install libxml-parser-perl**

2. Installation de l'archive glade2.12.2.zip

1. L'archive **glade2.12.2.zip** vous sera fournie par vos tuteurs respectifs durant la séance du Projet C.
2. Copiez cette archive sous le répertoire *home*.
3. Décompressez l'archive à l'aide de la commande :

unzip glade2.12.2.zip

4. Pour l'installation, ouvrir le dossier *glade-2.12.2*:

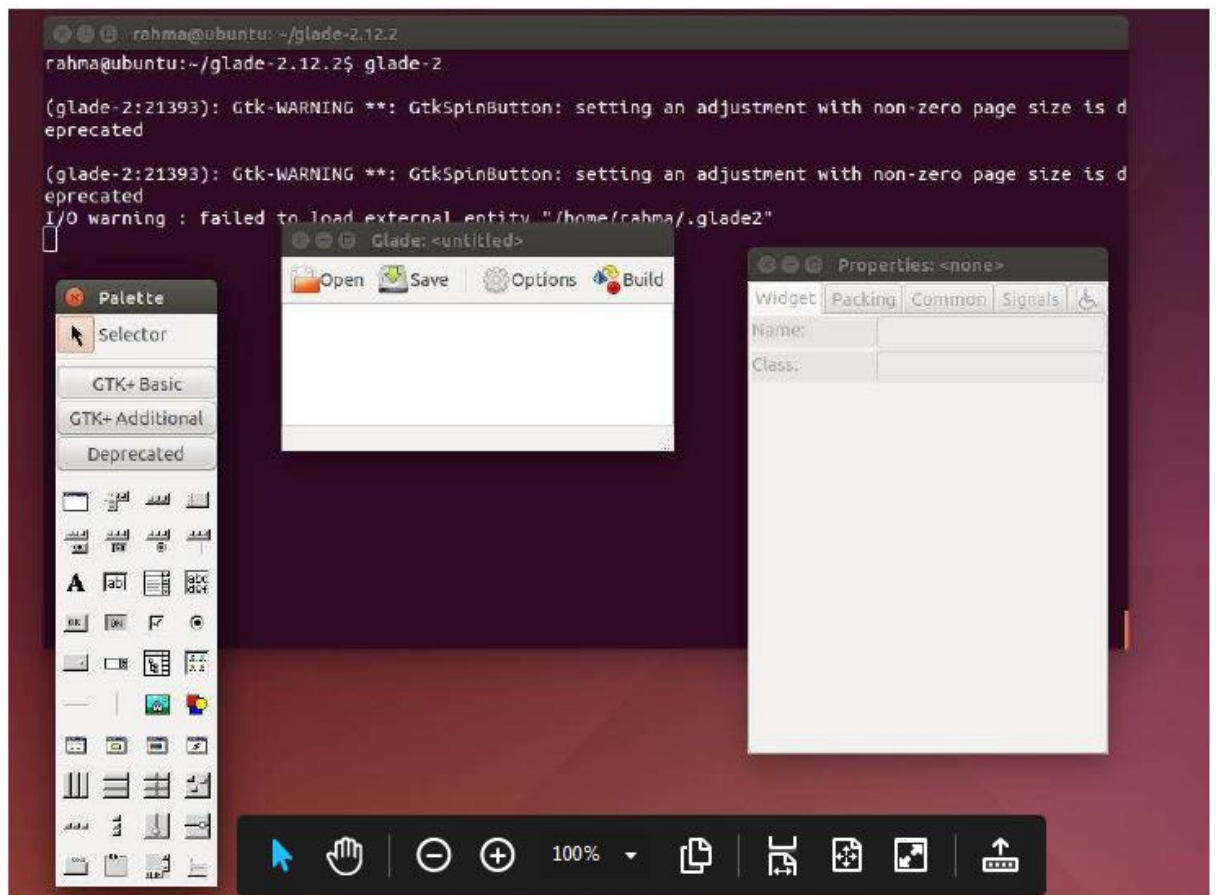
cd glade-2.12.2

5. Exécuter la commande :

./configure ;make ;sudo make install

✓ Lancement de Glade2

Pour lancer l'outil Glade 2, il suffit d'exécuter la commande **glade-2** :



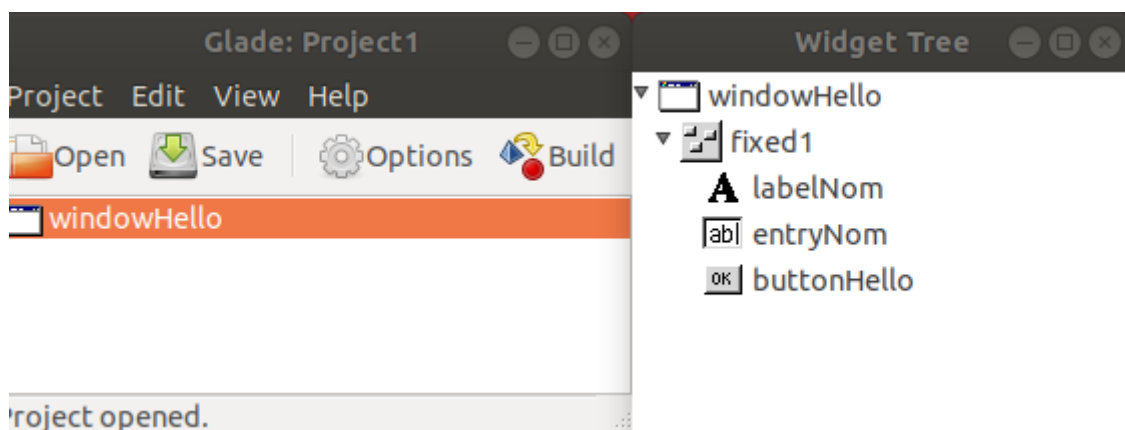
Présentation

1. L'interface glade

L'interface glade contient trois fenêtres :

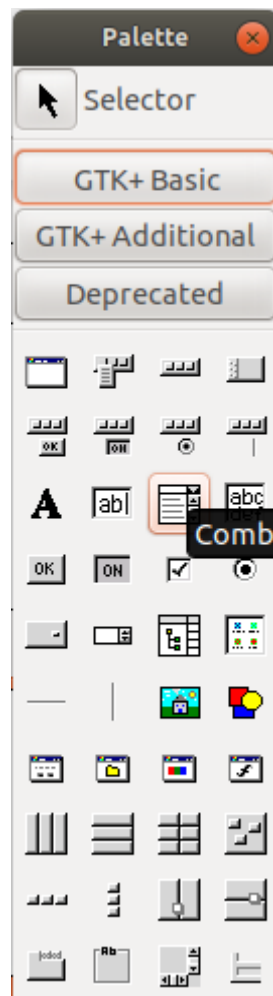
1.1 La fenêtre principale (de hiérarchie)

Elle permet de mieux visualiser la structure de votre projet. On peut distinguer les liens de parenté entre les différents composants graphiques (widgets).



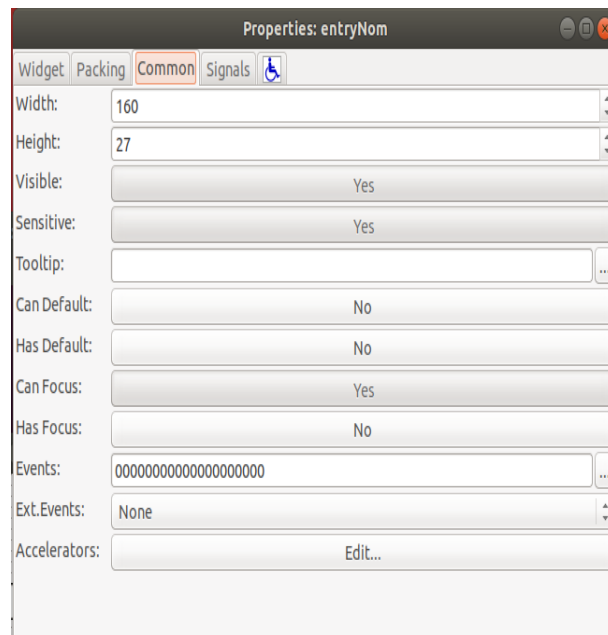
1.2 La palette de widgets

Tous les Widgets sont disponibles dans GTK, classés par catégorie. Il suffit de cliquer sur l'icône du Widget souhaité puis de cliquer à l'endroit voulu pour le placer



1.3 La fenêtre de propriété

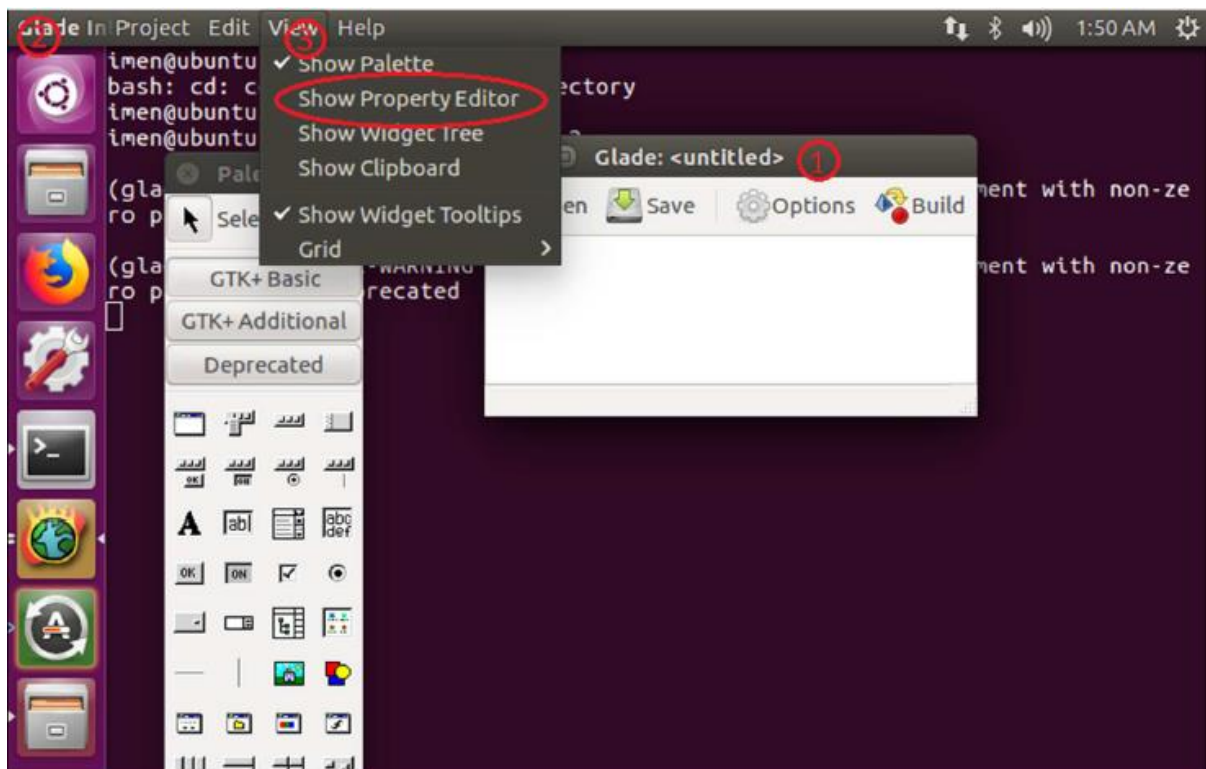
Elle permet de modifier les paramètres du Widget sélectionné, tel que sa taille, son étiquette, sa bulle d'aide, etc. Elle permet également la gestion des signaux par simple sélection de l'action désirée.



Remarque : Parfois l'une des trois fenêtres Glade-2 se ferme, supposons qu'il s'agit de la fenêtre

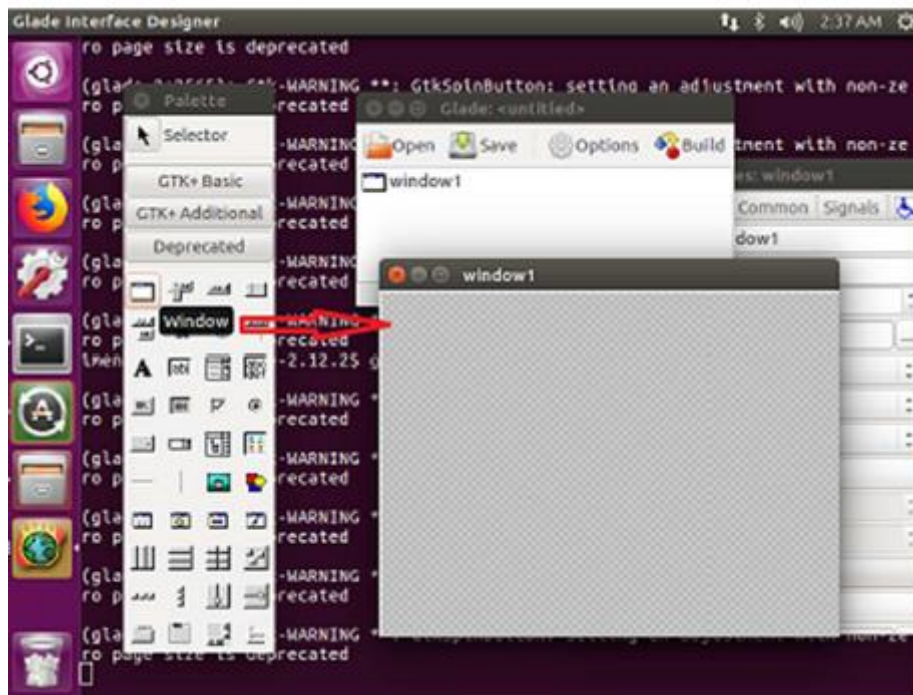
de propriétés. Pour la rouvrir, il suffit de procéder comme suit :

1. Cliquer sur le menu de la fenêtre principale
2. Cliquer sur le menu **Glade**
3. Cliquez sur le sous-menu **View** et choisir "**Show Property Editor**"



1.4 Comment créer une nouvelle fenêtre :

Pour créer une nouvelle interface de notre application, on choisit dans la palette des widgets le composant **Window** : une feuille blanche s'ouvre, dans laquelle nous pourrions ajouter les composants graphiques de notre interface.



2. GTK+

2.1 Les widgets

En GTK+, on appelle les objets graphiques classiques, comme les fenêtres, les boutons et les labels, des **widgets**.

Pratiquement ces objets sont de type `GtkContainer` et héritent de `GtkWidget` et peuvent contenir un ou plusieurs Widgets. Par exemple, une fenêtre peut contenir un bouton qui contient un label.



2.2 Les signaux

Les actions que l'utilisateur réalise sur l'interface graphique (clic sur un bouton, destruction d'une fenêtre,) sont récupérées par une boucle événementielle implémentée dans GTK+.

L'objet graphique (Widget) concerné par l'évènement émet un signal qui sera ensuite connecté à une fonction de traitement (callback). Il suffira ensuite au programmeur de la compléter selon ses besoins.

Partie 2 : Etude de cas : « Création d'une interface graphique qui permet à l'utilisateur de saisir son nom. »

Glade est l'un des outils les plus performants qui facilite la création des applications interactives avec GTK+. Voilà la démarche à suivre pour développer une application :

- 1-Utiliser Glade pour concevoir l'interface graphique
- 2-Générer le code source
- 3-Ajouter le code métier nécessaire pour le fonctionnement de l'interface
- 4-Compiler le projet.
- 5-Lancer l'exécutable

Dans ce tutorial on va voir un premier exemple. Notre but est de créer une interface graphique qui permet à l'utilisateur de saisir son nom :



L'appuie sur le bouton « Say Hello » permettra de modifier le label « Hello » selon le nom saisi :



1. La conception de l'interface avec Glade

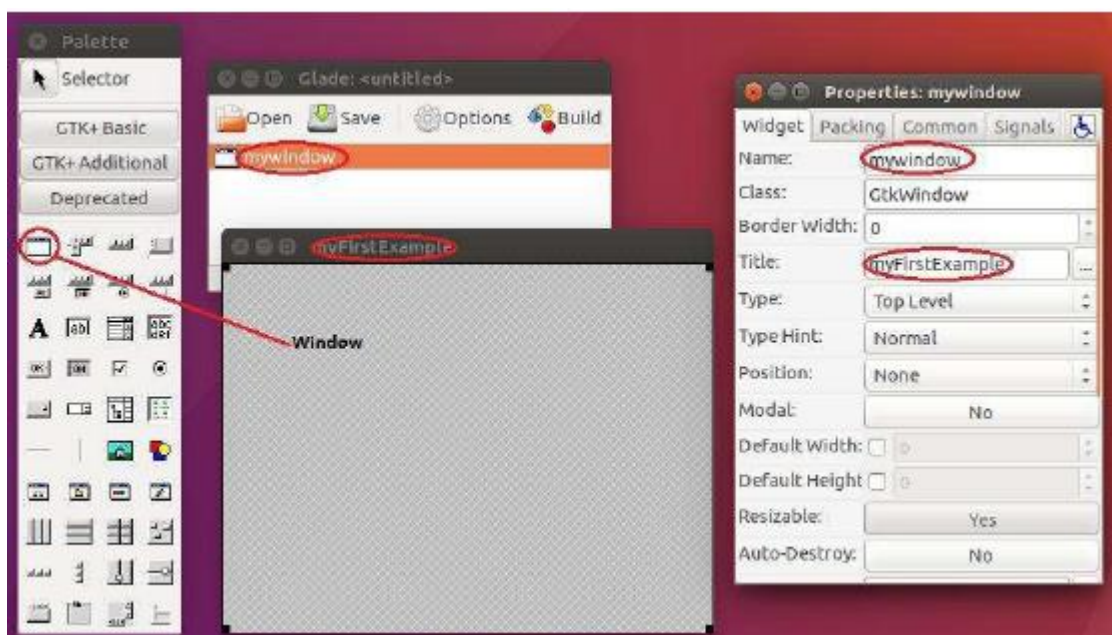
1.1 Création de la fenêtre principale

En lançant l'outil Glade, une fenêtre principale s'ouvre. On commence par ajouter un « window ». Ensuite, cliquez sur « window » et faites 'un drag and drop' dans la feuille blanche. Nous disposons à présent d'un widget de type GtkWindow* .

Dans la fenêtre **propriétés** on peut accéder aux différentes propriétés de notre widget et les modifier. Nommez votre window « myWindow ».

Dans le programme C les widgets sont accessibles à travers **leurs noms**.

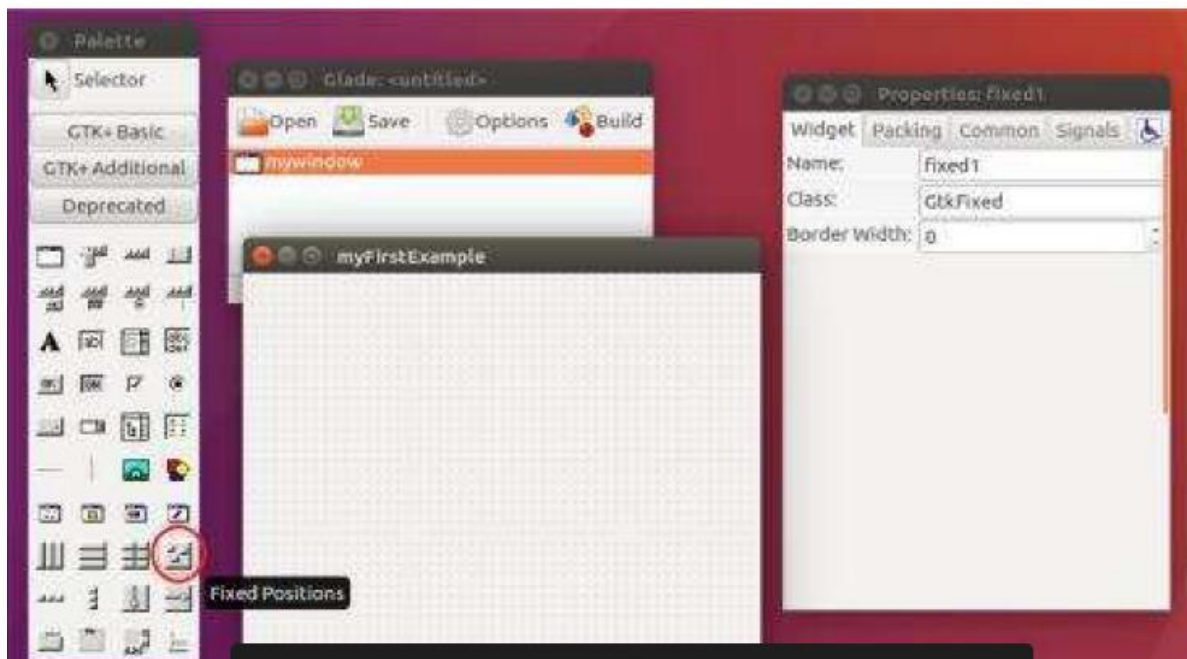
Le titre (dans window Title) sera myFirstExample. Vous pouvez découvrir les autres propriétés à modifier.



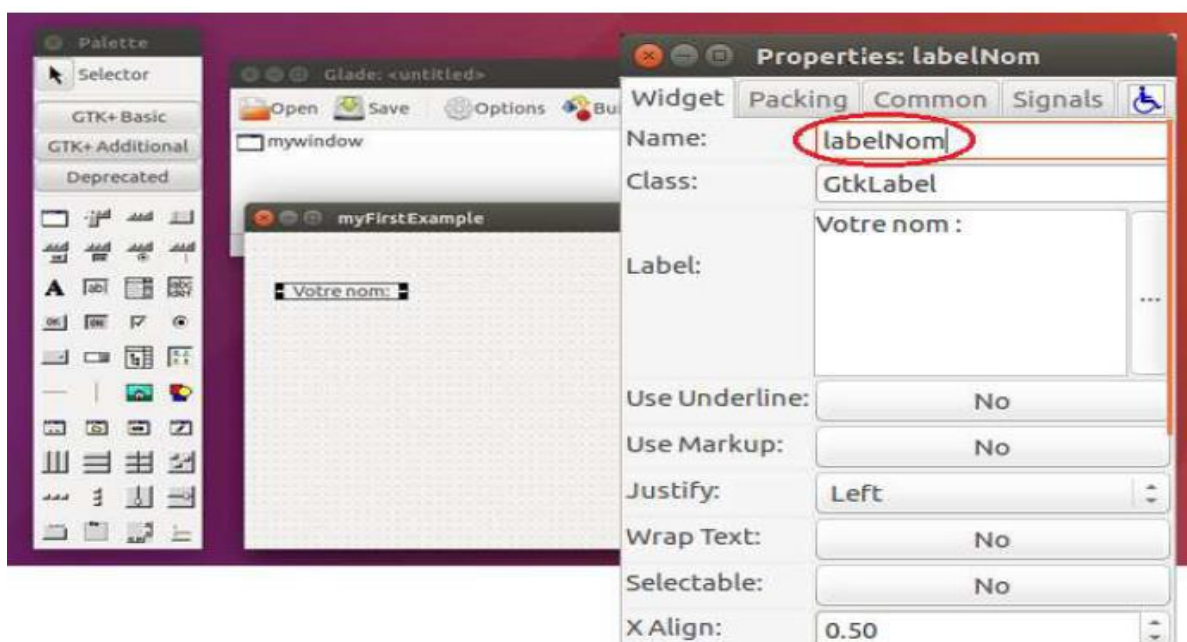
1.2 Ajout des widgets

Pour ajouter des widgets, il suffit de cliquer sur l'objet désiré dans la palette puis cliquez dans la fenêtre principale.

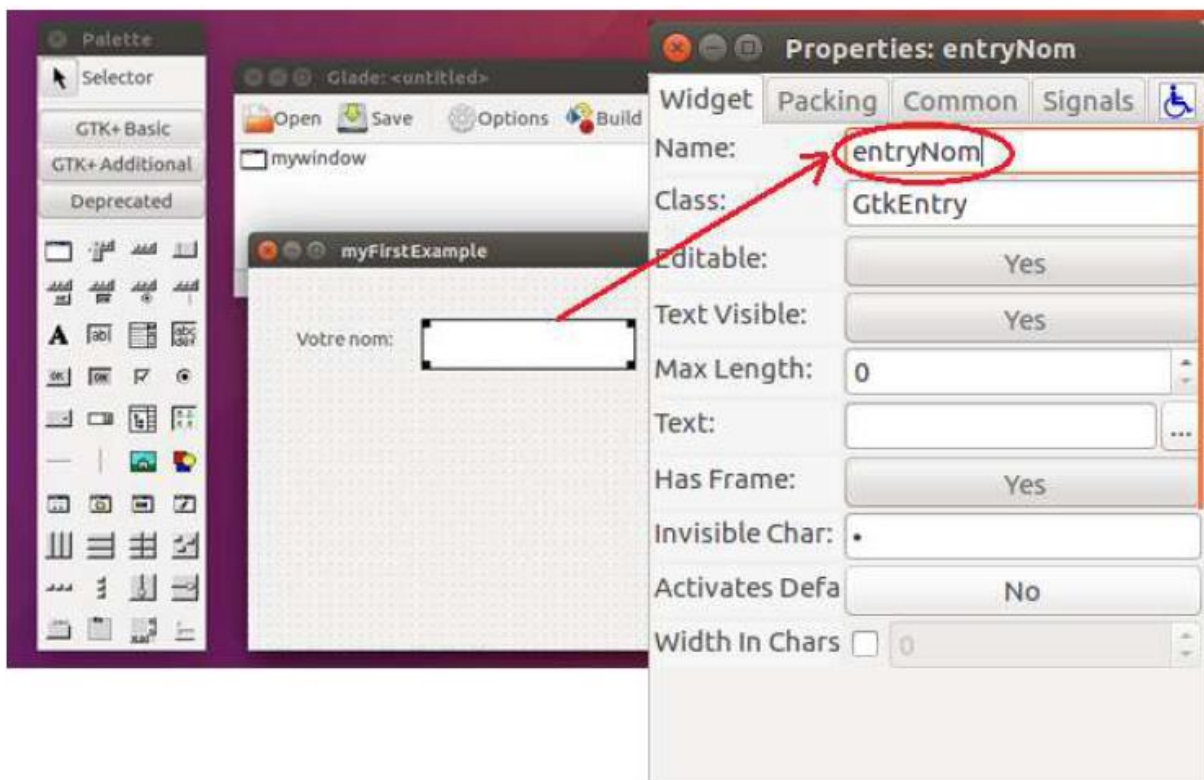
Remarque : Il est recommandé de commencer par le widget *Fixed Positions* qui facilite le positionnement des autres objets et en l'absence duquel, tout autre composant graphique ajouté à la fenêtre (bouton, label, ...) s'étalera sur toute la fenêtre (occupera tout l'espace réservé à la fenêtre).



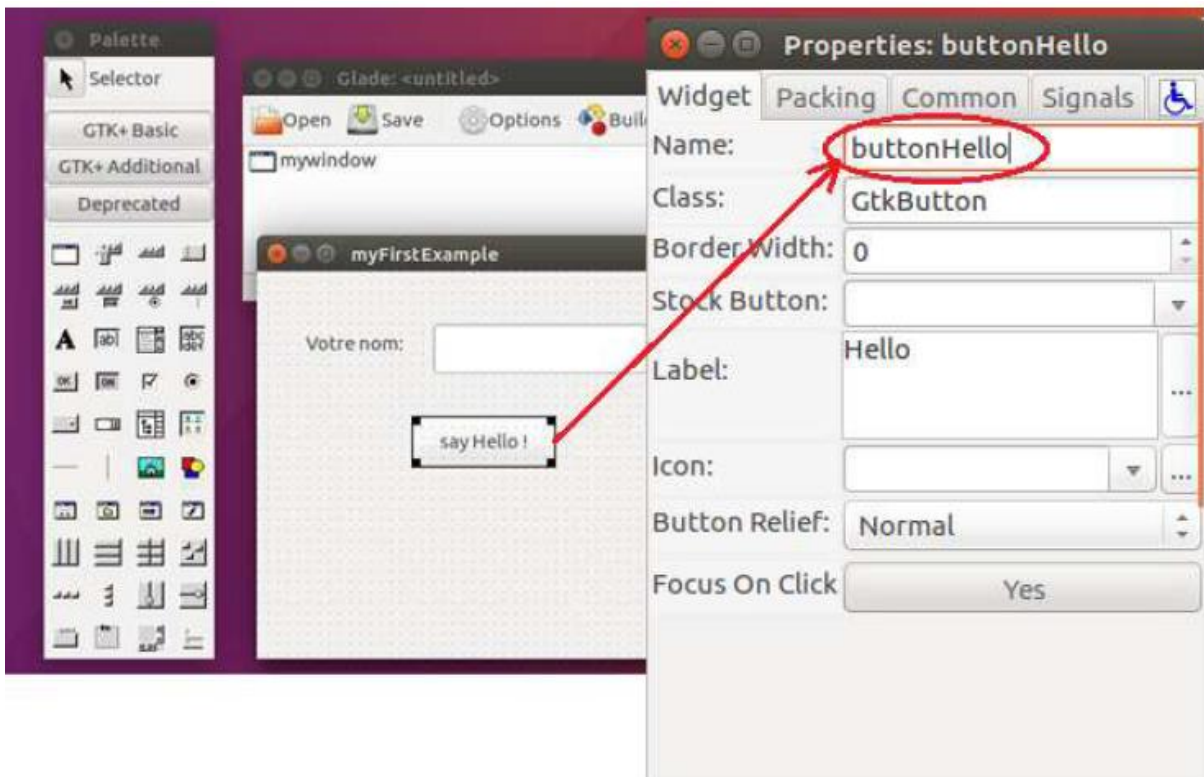
Ajoutez maintenant un widget de type *GTKLabel* dont le nom est *labelNom*:



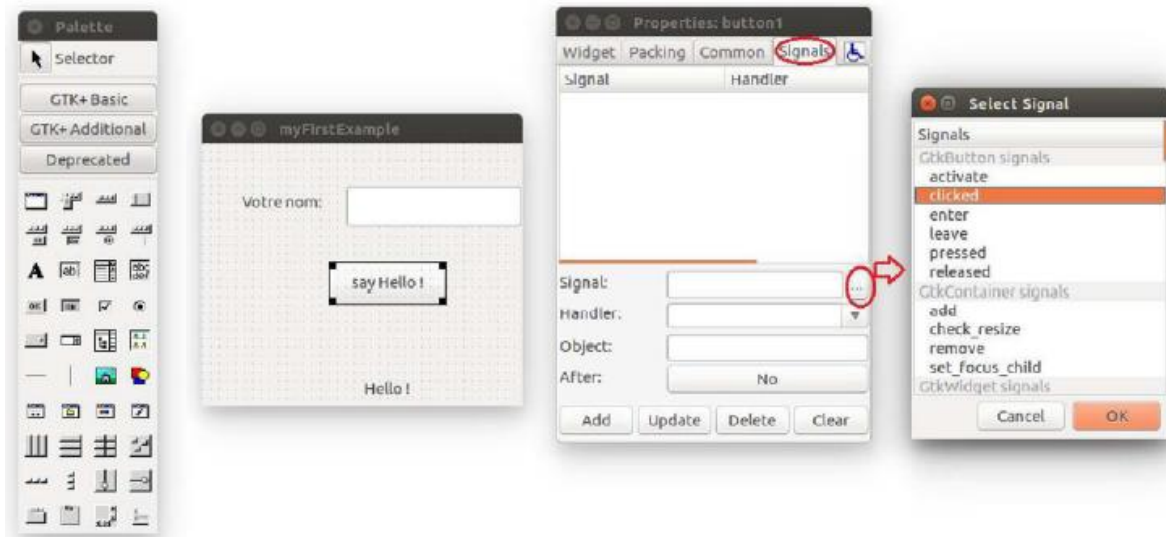
Ajoutez un widget de type **GTKEntry** dont le nom est **entryNom**:



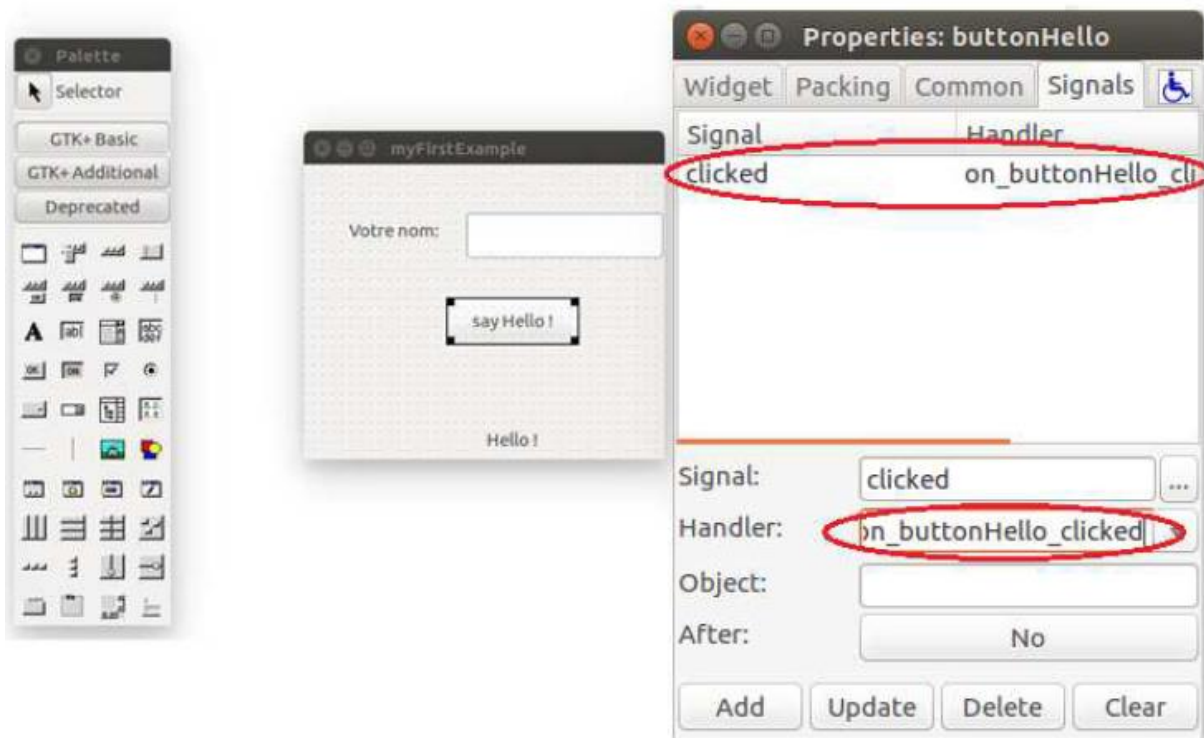
Ajoutez un widget de type **GtkButton** dont le nom est **buttonHello**:



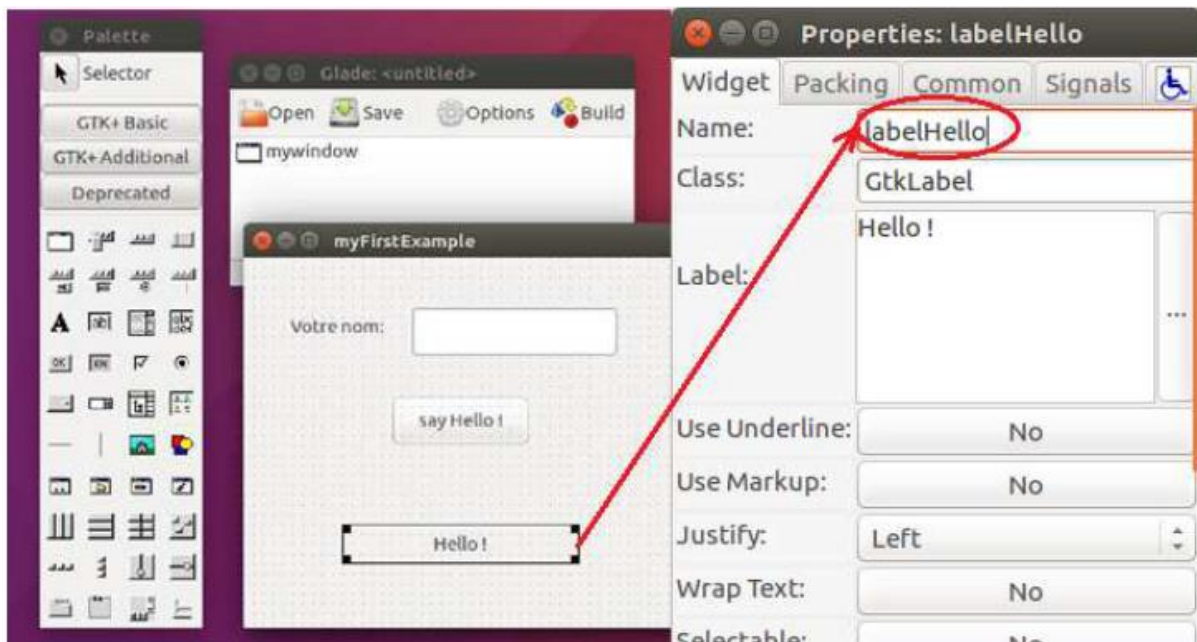
Ajoutez un signal à ce bouton comme suit :



Cliquez sur **Add** :



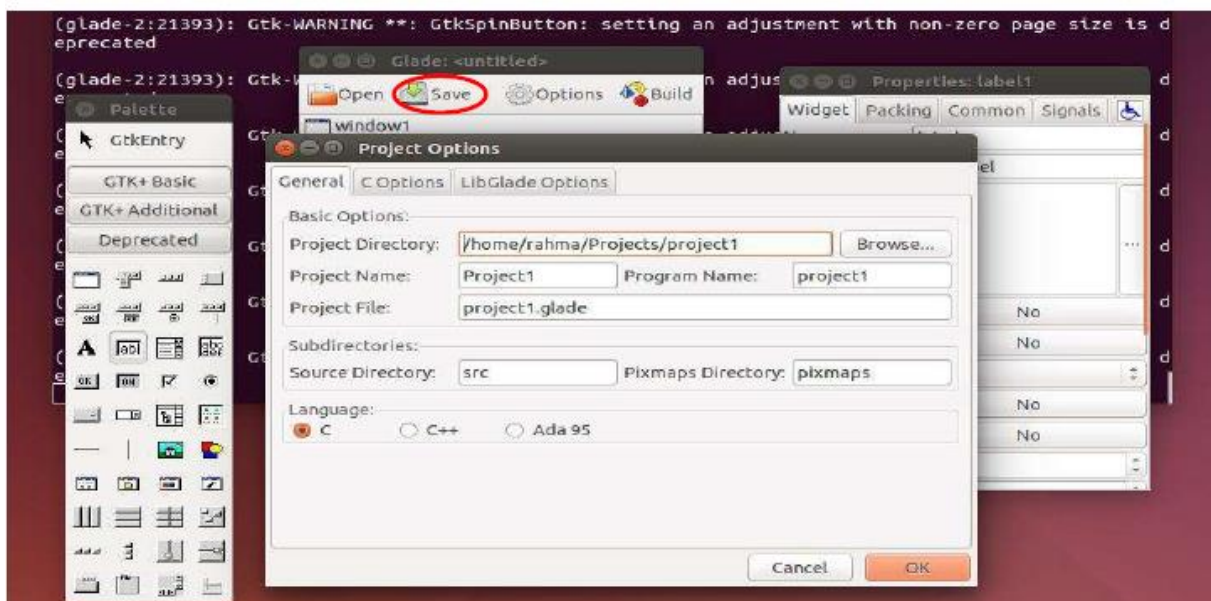
Enfin ajoutez un autre widget de type **GTKLabel** dont le nom est **labelHello**:



Remarque : Les noms affectés aux différents composants graphiques doivent impérativement être significatifs (*windowHello*, *labelNom*, *labelHello*, *entryNom*, *buttonHello*, ...).

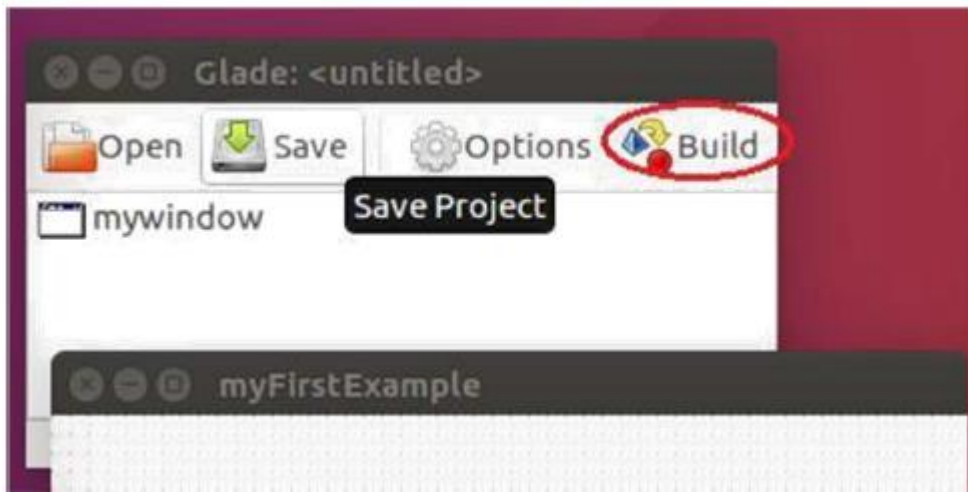
En effet, les noms affectés par défaut (*label1*, *label2*, *entry1*, ..., *entry50*) ne permettent pas d'identifier le composant en question dans le code, et ceci pose un problème lors de l'intégration de tout le projet.

Sauvegarder votre travail :



2. Générer le code source

Après le sauvegarde du projet , générez le code source en cliquant sur **Build** (ou générer) :



S'il s'agit du premier projet, il faut installer **autoconf** avec la commande :

sudo apt-get install autoconf

Vous pouvez maintenant accéder au dossier du projet

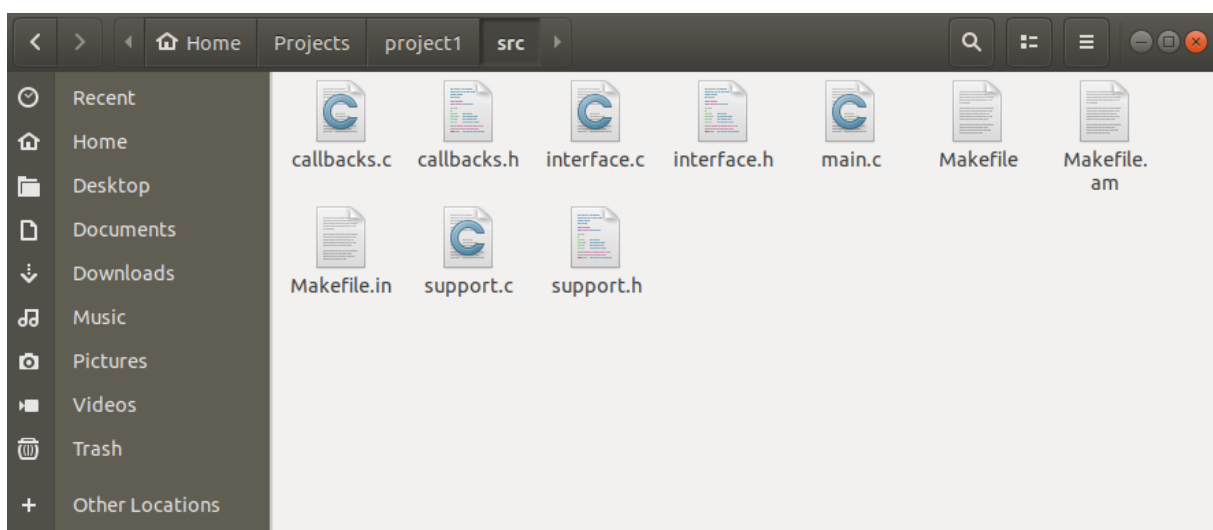
/home/<user>/Projects/project1

et exécuter la commande

./autogen.sh

Le script **autogen.sh** permet de simplifier la construction du projet, car il réalise toutes les opérations nécessaires de manière automatique. Les paquets **automake** et **autoconf GNU** sont utilisés pour simplifier la gestion de la configuration, la compilation et l'installation de votre application. Ces outils aident dans la gestion des projets de logiciels . Glade génère tous les fichiers nécessaires conformément à GNU.

Dans le sous-répertoire appelé **src** vous pouvez remarquer la création des fichiers d'entêtes et des fichiers sources.



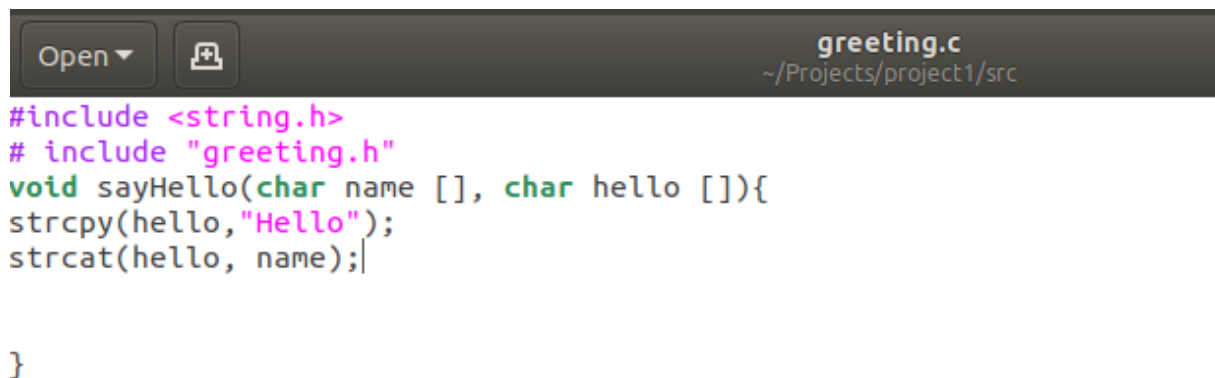
Module	Signification
support.c/support.h	On y trouve la fonction <code>lookup_widget()</code> qui permet de récupérer tout Widget grâce à son étiquette.
interface.c/interafce.h	Création de tous les Widgets avec leurs signaux associés. Une fonction par fenêtre est créée.
callbacks.c/callbacks.h	Les fonctions de traitement prête à être implémentée par le programmeur.
main.c	Appelle les fonctions de l'interface.

3. Ajouter le code métier

On va ajouter le module **greeting.h** / **greeting.c** sous le dossier **/home/<user>/Projetscs/project1/src**

Implémenter la fonction **void sayHello(char nom[],char hello[])** qui permet d'ajouter à la chaîne « Hello », le nom passé en paramètre et de sauvgarder la chaîne résultante dans la variable *hello* .

greeting.h :



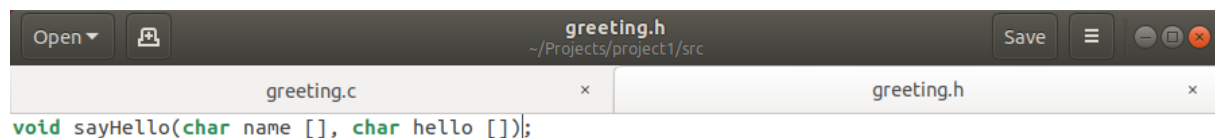
```

#include <string.h>
# include "greeting.h"
void sayHello(char name [], char hello []){
strcpy(hello,"Hello");
strcat(hello, name);

}

```

greeting.c :



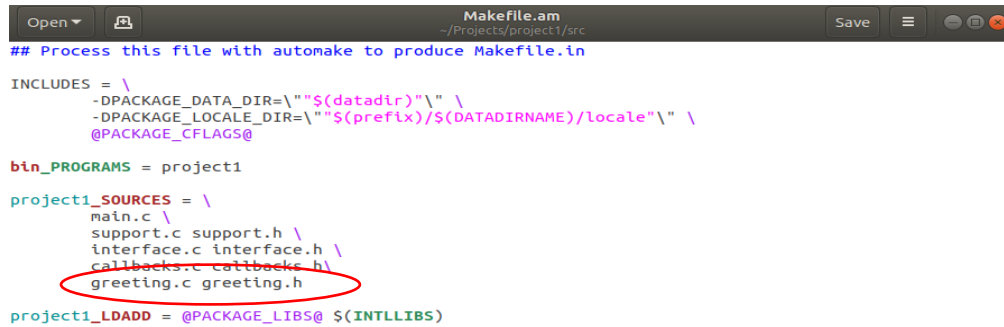
```

void sayHello(char name [], char hello []);

```

Maintenant, compilez le code pour corriger les éventuelles erreurs.

Ensuite, éditez le fichier Makefile.am (**/home/<user>/Projetscs/project1/src**) comme suit :



```

Makefile.am
~/Projects/project1/src
Save

## Process this file with automake to produce Makefile.in

INCLUDES = \
    -DPACKAGE_DATA_DIR=\"${datadir}\" \
    -DPACKAGE_LOCALE_DIR=\"${prefix}/${DATADIRNAME}/locale\" \
    @PACKAGE_CFLAGS@

bin_PROGRAMS = project1

project1_SOURCES = \
    main.c \
    support.c support.h \
    interface.c interface.h \
    callbacks.c callbacks.h \
    greeting.c greeting.h

project1_LDADD = @PACKAGE_LIBS@ $(INTLLIBS)

```

Maintenant, régénérez les fichiers makefile en appelant le script `./autogen.sh`

Il vous reste maintenant que de modifier le module **callbacks.h** / **callbacks.c** pour implémenter le signal du bouton « *sayHello* ».

Implémentation du signal « `on_buttonHello_clicked` »

Comme cité auparavant, Les widgets (Window gadget) sont des objets à la base de la programmation graphique en GTK+ (GtkWidget). Tout objet graphique hérite des propriétés et des fonctions relatives au Widget.

Afin de récupérer un message texte à partir des zones de saisie de texte (entry) ou d'écrire un message texte vers une zone d'étiquette (label), vous devriez accéder à ces widgets graphiques moyennant des pointeurs sur GtkWidget.

Chaque objet graphique de l'interface graphique (window, entry, label, bouton, etc...) doit lui correspondre un pointeur afin de y accéder. Ces pointeurs doivent être déclarés dans les fonctions auto-générées dans le module **callbacks.h** / **callbacks.c**. Dans notre cas, on a aussi le signal "clicked" du composant `buttonHello` qui est connecté à une fonction nommée `on_buttonHello_clicked()`. Pour que notre interface puisse fonctionner, il faut donc écrire le code de cette fonction (en d'autres termes, la fonction qui sera appelée lorsque le signal correspondant sera émis par le composant auquel elle est connectée). Son rôle sera de récupérer les valeurs saisies par l'utilisateur et d'afficher le text de `labelHello`.

Donc, suite à l'événement du clic sur le bouton `buttonHello`, la fonction `on_buttonHello_clicked` est appelée. On va la modifier de telle sorte qu'elle implémente les opérations suivantes :

✓ Récupérer le texte saisi dans *entryNom*

✓ Modifier le texte du *labelHello*

Démarche à suivre

1- Dans le module **callbacks.h** / **callbacks.c**, modifiez l'entête de la fonction `on_buttonHello_clicked` comme suit :

*void on_buttonHello_clicked (GtkWidget *objet_graphique, gpointer user_data)*

2- Dans **callbacks.c** et au niveau de la fonction `on_buttonHello_clicked`, déclarez deux pointeurs sur **GtkWidget**:

GtkWidget input ;*

GtkWidget output ;*

3- Déclarez deux chaînes de caractères **nom** et **hello** dans lesquelles on va sauvegarder les valeurs de **input* et **output* et qui vont servir comme paramètres pour notre fonction métier **sayhello(nom,hello)**

4- Liez les pointeurs **input* et **output* aux objets graphiques correspondants :

input = lookup_widget(objet_graphique, "entryNom") ;

output = lookup_widget(objet_graphique, "labelHello") ;

5- Récupérez le texte saisi dans **entryNom**

strcpy(nom,gtk_entry_get_text(GTK_ENTRY(input)));

6- Modifiez la chaîne **hello** :

sayHello(nom,hello);

7- Modifiez le texte de **labelHello** :

gtk_label_set_text(GTK_LABEL(output),hello);

3. Compilation

Sous-le dossier **/home/<user>/Projets/project1**, exécutez la commande :

make ; sudo make install

4. Exécution du projet

S'il n'y a pas des erreurs de compilation on peut lancer l'application, exécutez la commande :

project1