tutoriel glade

Eddy Ahmed

Ce tutoriel a pour but de fournir un manuel étape par étape pour les développeurs souhaitant écrire des applications GNOME en utilisant Glade. Il vous faut :

- les bibliothèques GNOME
- Glade
- quelques connaissances de base en langage C

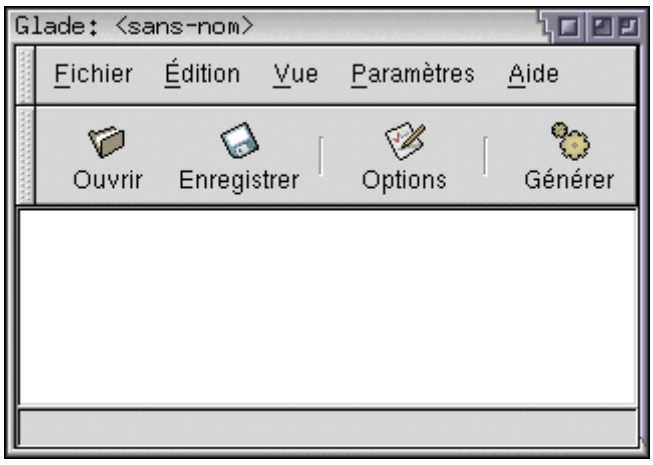
Vous allez faire une application " Bonjour le monde ! " modifiée comme ceci :

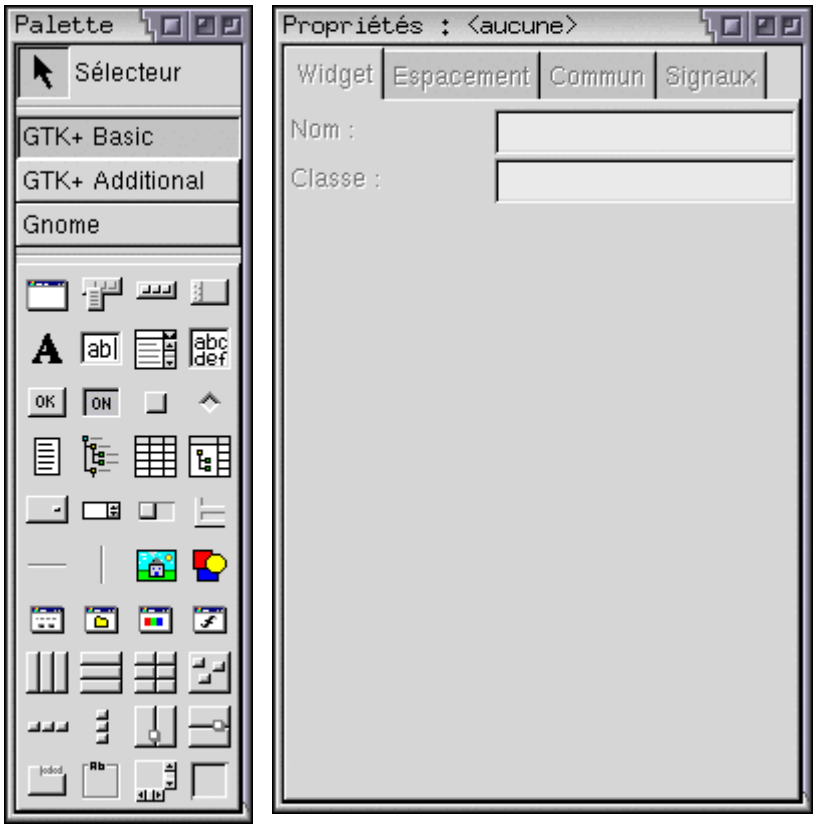
Pourquoi utiliser Glade ?

Glade permet au développeur de concevoir visuellement une application de manière rapide et efficace pour ensuite se concentrer sur l'implémentation du programme plutôt que de s'épuiser sur la création de l'interface utilisateur.

Comment ?

1) Démarrez Glade. Vous devriez voir les trois fenêtres suivantes :





La première fenêtre contrôle votre projet. La palette vous permet d'ajouter des widgets à votre application et la fenêtre des propriétés permet de modifier les propriétés d'un widget et de définir les signaux (nous y reviendront un peu plus tard).
2) La première chose dont nous allons avoir besoin est d'une surface sur laquelle ajouter des widgets, alors commençons par créer une fenêtre en cliquant sur l'icône fenêtre de la palette.

 icône "fenêtre" de la palette

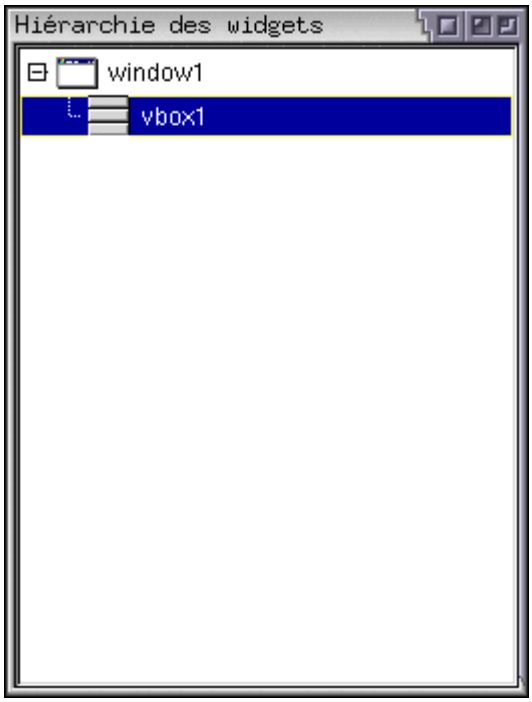
Une nouvelle fenêtre apparaît, prête à être modifiée. Remarquez les propriétés de la fenêtre, en particulier que le nom de la fenêtre est "*window1*". Modifions le titre de cette fenêtre qui est répercuté dans la barre de titre par quelque chose de plus descriptif. Remplacez simplement le titre "*window1*" dans l'onglet Widget de la fenêtre de propriétés par quelque chose comme "*Mon Appli GNOME* ".
3) Bon, donc la fenêtre est préparée, ajoutons maintenant quelques boutons et étiquettes... mais attendez ! Nous ne pouvons pas le faire tout de suite... Les widgets dans GNOME sont empaquetés. Empaqueter les widgets peut ressembler à un fardeau au début mais il s'agit en fait d'une méthode ingénieuse permettant pleins de choses formidables comme par exemple le redimensionnement automatique. Quand un utilisateur redimensionne une application nous aimerions que les widgets de la fenêtre soient redimensionnés pour profiter d'un espace plus important ou rétrécissent pour convenir à une fenêtre nouvellement dimensionnée. Empaqueter permet que cela se fasse automatiquement et dispense le développeur d'écrire du code pour le redimensionnement. Empaqueter se fait en créant des *boîtes* ou des *tables*. Ces widgets son invisibles, c'est à dire qu'il ne peuvent être vu à l'exécution bien qu'ils aient un effet direct sur l'application.
Si vous observez à nouveau l'application que nous allons créer, vous pouvez voir que l'on peut distinguer trois rangées : l'étiquette en haut, la saisie de texte au milieu et les deux boutons en bas.

Nous allons donc créer une boîte verticale avec trois rangées. Cliquer sur l'icône de la *boîte verticale* puis cliquez n'importe où dans la fenêtre que nous avons créé précédemment.

 icône "boîte verticale" de la palette.

Spécifiez que vous voulez créer trois rangées et cliquez sur "OK". La fenêtre est maintenant divisée en trois rangées, prêtes à recevoir des widgets supplémentaires.

À ce stade, vous pouvez ouvrir la fenêtre de *hiérarchie des widgets* en sélectionnant "*montrez l'arborescence des widgets*" du menu "*vue* ".

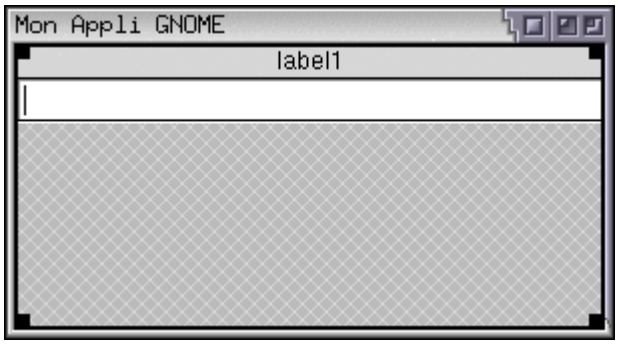


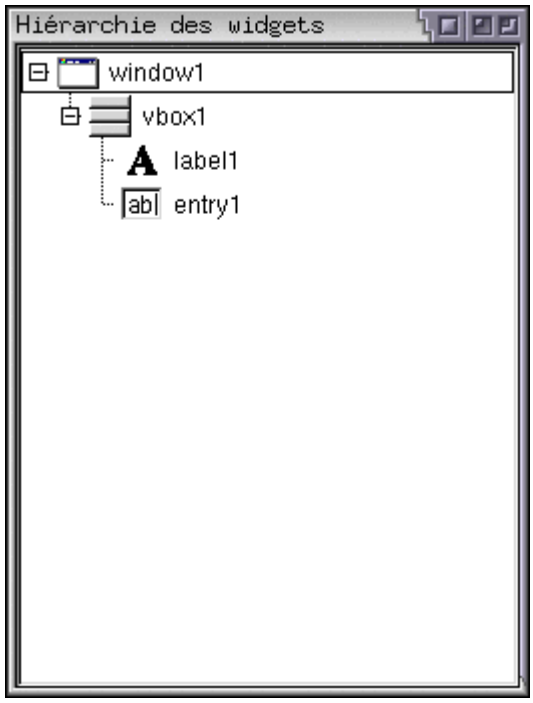
Cela va vous permettre de visualiser clairement la hiérarchie de vos widgets et de les sélectionner facilement.

Nous pouvons maintenant ajouter simplement les widgets "étiquette" et "saisie de texte" dans les premières et secondes rangées en cliquant sur l'icône "*étiquette* " de la palette puis sur la première rangée de la fenêtre et procéder d'une manière similaire pour disposer le widget "*saisie de texte*" dans la seconde rangée

 icônes "étiquette" et "saisie de texte" de la palette

La fenêtre de votre application et celle correspondant à l'arborescence des widgets devraient maintenant ressembler plus ou moins à ça :





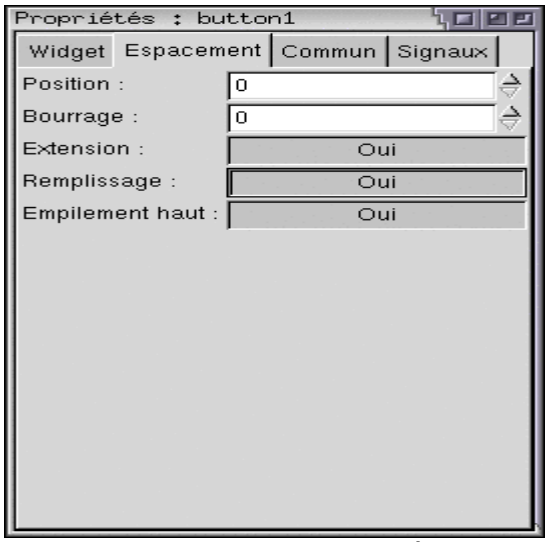
Avant que nous puissions ajouter des boutons dans la troisième rangée nous devons créer une boîte horizontale dans la troisième rangée de la boîte verticale. La nouvelle boîte horizontale aura deux colonnes, une pour chaque bouton. Donc, faisons le en cliquant sur l'icône de la boîte horizontale de la palette et en cliquant ensuite dans la troisième rangée de notre fenêtre.

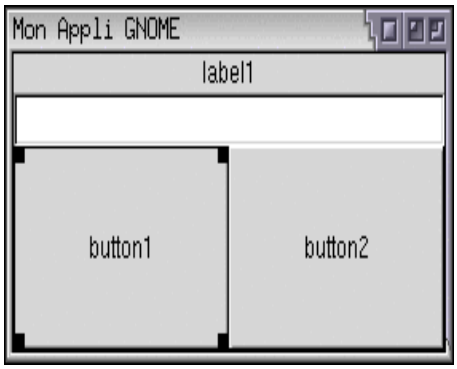
 icône "boîte horizontale" de la palette

Spécifiez deux colonnes puis cliquez sur "OK". La troisième rangée est maintenant divisée horizontalement en deux sections égales.
Le temps est maintenant venu de sauvegarder notre projet. Par défaut, Glade enregistre votre projet dans un répertoire "*Projets*" dans votre répertoire personnel. Renommez votre projet simplement en "*Bonjour*".
Retournons à nos boutons. Chacune des sections que nous avons créé dans la troisième rangée comportera un bouton. Alors, ajoutons un bouton à chaque section en cliquant sur l'icône bouton de la palette puis dans la première section. Effectuez une action identique pour la deuxième section.

 icône "bouton" de la palette

Bien, nos boutons sont donc bien là... mais ils n'ont pas une bonne allure. C'est parce que nous devons les empaqueter. Faites le en cliquant sur le premier bouton et dans la fenêtre des propriétés, sélectionnez l'onglet "*espacement* ". Accordez les propriétés "*extension*" et "*remplissage* " pour le bouton en en les positionnant sur "*oui*" par un simple clic. Vous devriez voir le premier bouton s'étendre un peu. Faites la même chose pour le deuxième bouton et vous devriez alors avoir un résultat plus appréciable :



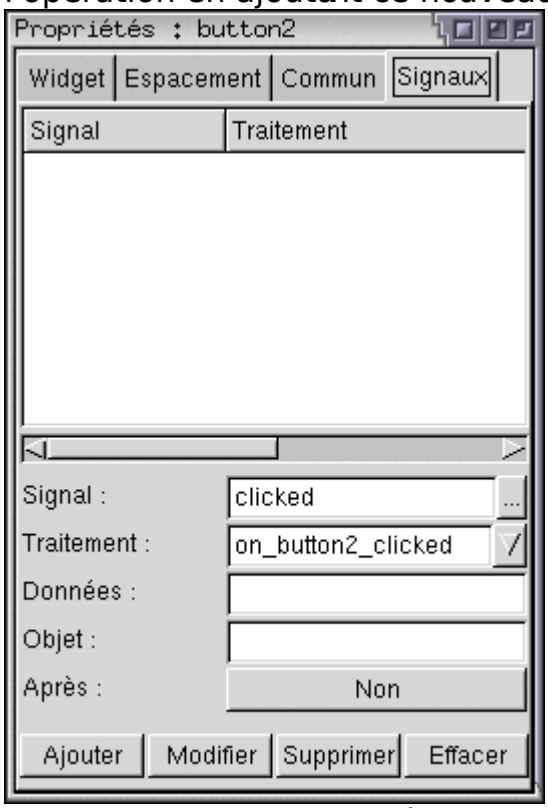


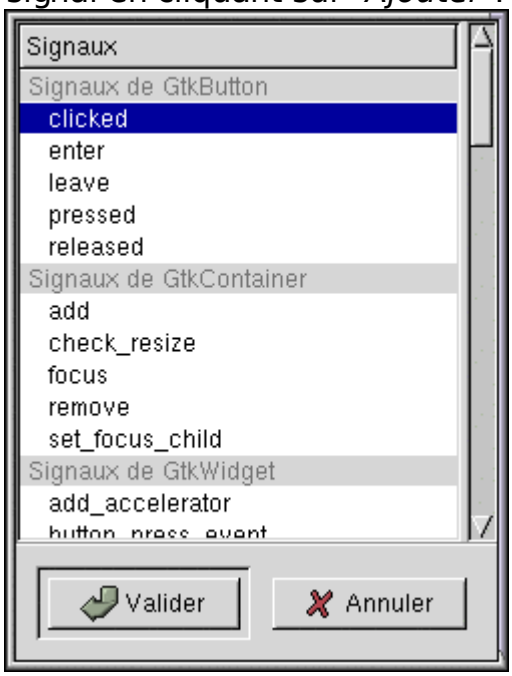
Nous pouvons encore améliorer l'allure en effectuant les changements suivants (souvenez vous que vous pouvez sélectionner facilement les widgets dans la fenêtre représentant leur arborescence) :

- changez la propriété "*homogène*" de *hbox1* en "*oui*"
- changez les propriétés "*largeur de bordure*" de *hbox1* et *vbox1* en "*4* "
- changez les propriétés "*espacement*" de *hbox1* et *vbox1* en "*4*"
- changez la propriété "*bouton prédéfini*" du *button1* en "*OK* "
- changez la propriété "*bouton prédéfini*" du *button2* en "*close* "

Une excellente chose lorsque vous développez avec Glade est que vous pouvez voir immédiatement à quoi votre application va ressembler... donc, modifiez les propriétés afin que cela ressemble à ce que vous souhaitez.
Une dernière propriété à modifier avant de commencer à éditer le code. Changez la propriété "*étiquette*" de *label1* en « *Entrez votre nom :* ».
Sans avoir écrit une seule ligne de code nous avons réellement créé une application fonctionnelle ! Demandez à Glade de générer du code source en cliquant sur "*Générer*" dans la barre d'outils ou en sélectionnant "*Générer le code source* " du menu "*Fichier*".
Si vous regardez maintenant dans le répertoire de votre projet (souvenez vous que nous l'avons enregistré dans `/home/nom_utilisateur/Projets/Bonjour/`) vous verrez tous les fichiers que Glade a créé.
Le véritable code source se trouve dans le sous-répertoire "src". Certains fichiers comme README, ChangeLog et d'autres que vous modifierez certainement lorsque vous développerez sérieusement d'autres applications. Laissons les dans l'état pour l'instant. Générez les Makefiles en exécutant **./autogen.sh** depuis ce répertoire dans votre terminal préféré. Une série de messages vont défiler durant les vérifications de votre environnement et la création des Makefiles appropriés. Une fois l'autogen.sh terminé nous pouvons vraiment construire notre application en exécutant [la commande] **make**.
Si tout se passe bien, vous devriez avoir une fichier "**bonjour**" exécutable dans le répertoire "src". Lancez le en tapant `./bonjour` .
Hop là ! Votre première application GNOME ! Testez le redimensionnement pour voir la qualité du système d'empaquetage.
Tout ceci est bien sympathique... mais notre application est encore

peu attrayante du fait qu'elle ne peut pas encore servir à quelque chose. Alors commençons donc la programmation !
Comme il s'agit essentiellement d'un tutoriel de présentation, nous n'allons pas faire de fantaisie. Nous devons faire en sorte que lorsqu'un utilisateur tape son nom dans la zone de saisie de texte puis clique sur "*Validez*", il soit salué par un "bonjour" suivi de son nom affichés dans l'étiquette. Et aussi que le bouton "*Fermer*" permette de quitter l'application.
Afin d'accomplir tout cela, vous devez comprendre le concept des signaux et des rappels [callbacks]. Un signal se produit chaque fois qu'un événement se produit comme un clic de souris. Donc nous devons dans un premier temps créer un traitement de signal pour chaque action qui devant entrainer un évènement. À chaque signal doit correspondre une fonction de rappel. La fonction est ce qui est réellement exécuté lorsqu'elle est appelée. Commençons donc par créer deux signaux, un pour chaque bouton ; commençons par le bouton "*Fermer* " dont l'implémentation se fera plus facilement. Sélectionnez le bouton "*Fermer* " (*button2*) et dans le fenêtre "*Propriétés* " sélectionnez l'onglet "*signaux*". Un bouton peut générer plusieurs signaux mais dans notre cas nous ne sommes intéressés que lorsqu'on le clique. Cliquez sur les trois points au niveau du champ signal et dans la liste sélectionnez "*clicked*" et validez. Dans le fichier ce champ correspond au nom de le fonction que vous modifierez pour cet événement précis. Terminez l'opération en ajoutant ce nouveau signal en cliquant sur "*Ajouter*".





Ajoutons un peu de code à l'un des fichiers nouvellement générés afin de voir agir notre événement. Par défaut, tous les rappels sont définis dans **callbacks.c** dans le répertoire "**src** " de votre projet. Avant de commencer à coder, prenez soin de générer le code source afin de mettre à jour les fichiers déjà créés. Ouvrez votre terminal et entrez dans le répertoire "src". Éditez le fichier "**callbacks.c**" avec votre éditeur préféré. Vous remarquerez qu'une nouvelle fonction attend que vous lui insuffliez de la vie ;

```
void on_button2_clicked (GtkButton *button, gpointer
```

```
user_data)
{
}
```
Ajoutons un peu de code pour voir agir notre événement. La fonction `g_print` est utilisée pour afficher vers la sortie standard, donc ajoutons du code pour signifier que le "button2" a été cliqué. Modifions la fonction de notre rappel ainsi :
```
void on_button2_clicked (GtkButton *button, gpointer
user_data)
{
g_print("Button2 clic !\n");
}
```
Enregistrez le fichier et, toujours dans le répertoire "src", tapez "make". Lancez l'exécutable (`./bonjour`) et cliquez sur le bouton "Fermer". Et là, "Button2 clic !" s'affiche dans le terminal !
Maintenant que l'on sait comment appeler notre événement assignons lui quelque chose de plus utile. Vous pouvez utilisez la fonction `gtk_main_quit` pour quitter votre application. Donc, tout ce que vous avez à faire est de remplacer le `g_print` par `gtk_main_quit();`
Votre bouton "Fermer" est maintenant pleinement fonctionnel !
Récapitulons ce que nous souhaitons que notre bouton "*Validez*" fasse. Après qu'un utilisateur ait tapé son nom dans la zone de saisie de texte, nous modifions l'étiquette pour y afficher "Bonjour Utilisateur !". Cela induit deux nouveau concepts : récupérer des données d'un widget et définir les propriétés d'un widget.
Les fonctions spécifiques aux widgets sont parfaitement documentés . Parmis d'autres choses, ces ressourcent documentent les fonctions spécifiques dont nous avons immédiatement besoin : celles des étiquettes et des saisies de texte.
La première chose que nous avons besoin est d'obtenir ce que l'utilisateur a tapé. Cela s'effectue avec la fonction `gtk_label_set_text()` . Cette fonction a aussi besoin comme arguments de pointeurs vers le widget approprié. Dans Glade cela s'accomplit avec la fonction `lookup_widget()`.
Maintenant que nous avons réunis les outils, mettons les en place. Pour commencer, il faut générer une ligne de traitement correspondant à notre bouton Validez. Sélectionnez ce bouton (button1). Dans la fenêtre des propriétés sélectionnez l'onglet signaux et ajoutez un signal "clicked" en laissant le nom de traitement par défaut. Générez le code source et vous devriez maintenant voir une fonction `on_button1_clicked` vide dans votre fichier callbacks.c.
La première chose que nous avons besoin de faire est de disposer des pointeurs pour les widgets que nous allons utiliser : label1 et entry1. lookup_widget renvoie un pointeur de type GtkWidget. Créeons deux pointeurs appellés label et entry correspondant à nos widgets. Le code ressemblera à cela :
```
GtkWidget * label = lookup_widget(GTK_WIDGET(button),
"label1");
GtkWidget * entry = lookup_widget(GTK_WIDGET(button),
"entry1");
```
Le premier paramètre de lookup_widget est un pointeur vers l'appel des données du widget, dans ce cas button (voir le premier paramètre de notre fonction rappel on_button1_clicked). Le second

paramètre est le nom du widget vers lequel nous souhaitons orienter le pointeur. Maintenant que nous avons nos widgets, utilisons les !
Avant de placer le texte de l'étiquette label1 nous allons avoir besoin de déclarer un chaine contenant ce que nous voulons que ça exprime. Souvenez vous que nous souhaitons d'abord que cela dise "Bonjour" donc du code C s'en chargera :

```
gchar output[50]= "Bonjour";
```

Nous venons là de créer un tableau de caractères appelée output de type gchar et placé dedans "Bonjour" avec un espace pour initialiser la première partie de notre message. Ensuite nous voulons concaténer le contenu de notre widget d'entré à cela. Nous accomplirons cela avec notre fonction gtk_entry_get_text qui nous renverra le texte contenu dans notre widget de saisie de texte. Quelque chose comme ça devrait convenir :

```
strcat(output, gtk_entry_get_text(GTK_ENTRY(entry)));
```

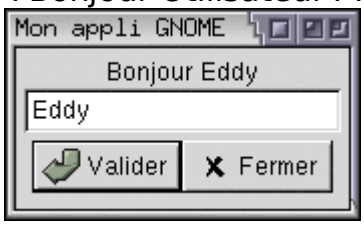
Souvenez vous que nous voulons passer un pointeur à `gtk_entry_get_text` . Donc nous devons opérer un cast sur le pointeur autrement vous obtiendrez des avertissements pendant le make.
Maintenant, tout ce que nous avons à faire est de mettre le texte de notre widget étiquette avec notre nouvelle chaine en utilisant `gtk_label_set_text` et c'est terminé !

```
gtk_label_set_text(GTK_LABEL(label), output);
```

Notre fonction complète est maintenant :

```
void on_button1_clicked(GtkButton *button, gpointer
user_data){
GtkWidget * label = lookup_widget(GTK_WIDGET(button),
"label1");
GtkWidget * entry = lookup_widget(GTK_WIDGET(button),
"entry1");
gchar output[50]="Hello ";
strcat(output, gtk_entry_get_text(GTK_ENTRY(entry)));
gtk_label_set_text(GTK_LABEL(label), output);
}
```

Terminé ! Exécutez un "make" dans le répertoire "src" et lancez votre application. Tapez votre nom dans le champ de saisie de texte et cliquez sur "Validez". Bonjour Utilisateur ! Pas mal...



Vous êtes maintenant sur la bonne voie pour développer votre propre application GNOME avec Glade. Même compte tenu de la simplicité de notre exemple, en réussissant ce tutoriel vous avez acquis des connaissances suffisantes pour vous lancer dans des développements plus complexes. Consultez les ressources Glade GTK+ et GNOME et n'ayez pas peur de vous lancer dans de nouvelles expériences. En espérant que cela vous a plu !