GenCodeC

version 2.00
Documentation française

Eric Totel

1 Introduction

MUIBuilder produit non pas un code directement compilable, mais une description du source final qui peut être adaptée (au prix d'un petit effort il est vrai) dans n'importe quel langage cible (tout du moins ceux qui existent sur l'Amiga et qui sont supportés par MUI !).

Pour que le code générique produit par **MUIBuilder** soit utilisable, il doit être traité par le générateur de code externe correspondant au langage cible choisi par l'utilisateur. `GenCodeC` est le module externe permettant la génération du langage C.

GenCodeC est **FREEWARE**. Toutefois je conserve le copyright sur mon travail. Vous pouvez modifier ce programme (les sources sont incluses) et en redistribuer votre propre version tant que :

1. Vous specifiez que je suis l'auteur originel de ce programme.
2. Vous m'envoyez une version de votre programme AVANT de le distribuer.
3. Cette documentation (contenant mon nom) est distribuée avec votre programme. (Vous pouvez y ajouter ce que vous voulez)

2 Comment est écrit GenCodeC ?

De manière à ce que chacun puisse réaliser un générateur de code qui correspond à ses besoins, vous trouverez inclus dans cette archive les sources du générateur de code pour le langage C.

Le rôle de ce programme est de traiter les fichiers temporaires générés par **MUIBuilder** dans T:. Dans ce but il utilise les fonctions qui sont mises à sa disposition dans la `MUIBuilder.library`. Ces fonctions sont commentées dans les autodocs.

3 Utilisation

Deux problèmes se posent au niveau de l'utilisation du générateur de code : d'une part celle du programme lui-même, et d'autre part du source qu'il crée. Ce sont ces deux points que nous allons aborder ici.

3.0.1 Utilisation de GenCodeC

`GenCodeC` est un programme prévu pour être lancé depuis un shell. Il est normalement exécuté par **MUIBuilder** lorsque vous demandez la génération d'un source, pour peu que vous ayez effectivement sélectionné le langage C dans le panneau de configuration de **MUIBuilder** !

Si vous le lancez depuis un shell sans que **MUIBuilder** soit actif, il n'y aura probablement aucun fichier temporaire de **MUIBuilder** dans T:, donc `GenCodeC` ne générera absolument rien. Aucun paramètre n'est nécessaire à son exécution, tous les paramètres étant passés par **MUIBuilder** dans ses fichiers temporaires.

3.0.2 Code généré

`GenCodeC` génère, pour un objet `foo` donné, appartenant à une application `App`

- un fichier .h contenant la définition de l'objet foo
- un fichier .c (dont vous aurez défini le nom dans **MUIBuilder**)[1] contenant une fonction `Createfoo` et retournant l'objet défini dans le fichier .h
- un fichier fooExtern.h contenant la définition des références externes vers des variables ou des fonctions de votre programme.

Deux fichiers de configuration nommés `H-Header` et `C-Header` permettent de configurer le code généré par **MUIBuilder** : en effet, ils sont copiés en entête respectivement des fichiers .h et .c dont il est question dans les lignes ci-dessus. Ainsi vous pouvez avoir les includes que vous désirez dans les sources générées par **MUIBuilder**.

La procédure de création se charge non seulement de la création de l'interface, mais également des notifications et de l'ouverture des fenêtres. L'exemple que vous trouverez dans cette documentation est donc tout à fait utilisable directement pour tester votre interface graphique.

3.0.3 Exemple d'utilisation

[1] si vous ne mettez pas une extension .c à votre nom de fichier, elle sera automatiquement ajoutée

```
#include <libraries/mui.h>

/* protos */
#include <clib/muimaster_protos.h>
#include <clib/alib_protos.h>
#include <clib/dos_protos.h>
#include <clib/exec_protos.h>

/*  Pragmas  */
#include <pragmas/muimaster_pragmas.h>
#include <pragmas/exec_pragmas.h>

/*  Ansi  */
#include <stdlib.h>
#include <stdio.h>

/* MUIBuilder */
#include "NONE.h"

struct Library * MUIMasterBase;

/* Init function */
static void init( void )
{
if (!(MUIMasterBase = OpenLibrary(MUIMASTER_NAME,MUIMASTER_VMIN)))
{
printf("Can't Open MUIMaster Library");
exit(20);
}
}

/* main function */
main()
{
struct ObjApp * App = NULL; /* Application object */
BOOL running = TRUE;
ULONG signal;

/* Program initialisation ( you need to write it yourself) */
init();

/* Create Application : generated by MUIBuilder */
App = CreateApp();

while (running)
      {
            switch (DoMethod(App->App,MUIM_Application_Input,&signal))
            {
            case MUIV_Application_ReturnID_Quit:
                  running = FALSE;
```

```
                    break;
                }
if (running && signal) Wait(signal);
        }
DisposeApp(App);
CloseLibrary(MUIMasterBase);
exit(0);
}
```

3.0.4 Objets non disponibles.

Je suis certain que je vais encore recevoir un nombre important de messages de personnes qui se sentent frustrées par le manque de certains objets MUI au niveau de MUIBuilder !!! Je vais donc prendre les devants des maintenant pour expliquer que ce n'est EN AUCUN CAS un problème. En effet MUI offre la possibilité d'ajouter dynamiquement des objets à une interface : Prévoyez un emplacement dans votre interface (un groupe vide fera parfaitement l'affaire) lorsque vous la créez sous MUIBuilder, et utilisez les possibilités qui vous sont offertes par MUI.

Ainsi, vous allez créer votre propre fonction de création d'objet (appelons la `CreateMyObject`), et l'utiliser de cette manière :

```
extern APTR CreateMyObject();

/* Utilisation du code de MUIBuilder */
App = CreateApp();

/* Ajout dynamique de l'objet que vous désirez */
DoMethod(App->GroupeVide, OM_ADDMEMBER, CreateMyObject());
```

4 Détails à propos du code généré

Certains détails à propos du code généré peuvent être intéressants à connaitre.

1. La structure générée dans le fichier header contient uniquement les labels des objets que vous avez définis comme générés au niveau de **MUIBuilder** (c'est-à-dire qu'ils apparaissent avec la lettre 'G' dans la fenêtre de création principale).
2. Le fichier fooExtern.h contenant les références externes vers des variables ou fonctions de votre programme va contenir des variables ou fonctions d'un type standard qu'il vous faudra modifier. Une fois cette modification faite, elle ne sera PAS effacée lors de la prochaine génération de code : `GenCodeC` ne génère que les extern qui n'existent pas déjà dans ce fichier.

3. Le fichier header utilisé pour la localisation DOIT s'appeler foo_cat.h.
4. Je vous conseille d'adapter dès maintenant les fichiers `C-Header` et `H-Header` à vos besoins : ce sont à mon avis d'importants fichiers de configuration, qui vous permettront d'adapter le source de **MUIBuilder** à vos options de configuration habituelles.

Résumé du contenu du document

Table des matières