LANGAGE SPL

Guide d'utilisation de l'interpréteur

I. Introduction

L'interpréteur **SPL** est capable de traduire un code en **SPL** en **C**, de le compiler en faisant appel à soit **TinyCC** (livré avec le programme) soit à **GCC** (Qui doit être installé par l'utilisateur). Le programme est disponible sous **Windows** et sous **Linux**.

Il est conseillé d'utiliser le programme en ligne de commande, soit grâce à un terminal sous **Linux** où bien avec cmd.exe sous **Windows**.

Le programme fonctionne en 4 grandes étapes :

- Dans un premier temps, il récupère le contenu du fichier contenant le code, et le décompose pour le structurer dans la mémoire (Sous forme de listes chainées).

- Dans un second temps, il génère à partir de la première structuration une nouvelle structuration contenant les fonctions et leurs attribues (nom, arguments, etc...) ainsi que les variables. Il en profite également pour vérifier que tout est correct.

- Puis le programme analyse enfin chaque ligne de chacune des fonctions, et vérifie que les opérations sont possibles.

- Enfin, le programme génère un code **C** à partir de ces listes chainées, et passe la main à un compilateur pour générer un exécutable.

II. Arguments indispensables

Fichier d'entrée

Il est nécessaire d'indiquer au programme quel fichier il doit traiter, il est donc obligatoire de l'indiquer comme argument d'entrée au programme.

Par défaut (Si aucun n'est indiqué), le programme utilisera comme fichier de sortie **out.exe** sous **Windows** ou out sous **Linux**, et le programme utilisera **TinyCC** comme compilateur. La création du fichier de log est également activée par défaut.

III. Arguments facultatifs

Mode verbeux (-v/-nv)

Le mode verbeux détaille plus amplement des différentes étapes de la compilation, sans toutefois rentrer dans les détails de traitement. Par défaut le mode verbeux est désactivé.

Voici un aperçu de son résultat :

```
C:\Users\Vuzi\Desktop\projet_windows>projet.exe -v pgcd.sp pgcd.exe
[Lancement du programme]

[Info] Debut de la sequence d'initialisation du programme...termine
[Info] Ouverture du fichier a traiter et debut du traitement
[Info] Suppression des commentaires...termine
[Info] Suppression des caracteres inutiles...termine
[Info] Passage en minuscule du code...termine
[Info] Fin de recuperation du code
[Info] Debut du premier parcours du code pour structuration
[Info] Fin du premier parcours du code pour structuration
[Info] Debut du second parcours du code pour recuperation des informations
[Info] Fin du second parcours du code pour structuration
[Info] Debut du troisieme parcours du code pour verification des informations et
 construction du code
[Info] Fin du troisieme parcours du code pour verification
[Info] Debut de la creation du code en C
[Info] Ouverture du fichier temporaire ou generer le code
[Info] Fermeture du fichier temporaire
[Info] Fin de la creation du code en C
[Info] Lancement de la compilation avec TinyCC
[Info] Fichier compile avec succes. Programme : pgcd.exe
[Info] Aucune erreur ou warnings durant la compilation
[Info] Compilation terminee, appuyez sur une touche pour quitter...
```

Log (-log/-no-log)

Le log garde une trace écrite de toutes les utilisations du programme. Le fichier utilisé est **log.txt** situé à la racine de l'exécutable. L'option **–no-log** permet de le désactiver.

Par défaut le log est activé.

Activer ou désactiver la compilation (-compil/-no-compil)

Il est possible de lancer le programme en lui indiquant de ne pas compiler le fichier. Le fichier de sortie sera alors un fichier texte contenant le code **C** traduit depuis le code en **SPL**. Le nom de fichier de sortie sera par défaut **out.c** sous Linux et Windows.

Si la compilation est activée (par défaut) alors le fichier **C** sera détruit après la compilation. Dans ce cas les noms de sortie par défaut seront respectivement out.exe et out pour **Windows** et pour **Linux**.

Choix du programme de compilation (-gcc/-tcc)

Il est possible d'utiliser **TinyCC**, ou bien **GCC** pour compiler le code traduit depuis le SPL. Sous Windows et Linux, **TinyCC** est livré avec le programme et permet de compiler sans avoir à installer quoi que ce soit. Il suffit d'utiliser l'option **–tcc** (Sélectionnée par défaut).

Utiliser GCC nécessite de l'avoir préalablement installé. Attention, sous **Windows** il est nécessaire d'avoir indiqué le répertoire d'installation de **GCC** dans les variables d'environnement (Ordinateur->Propriétés->Paramètre systèmes avancés->Variables d'environnement->Variables système, et ajouter l'emplacement de **GCC** avec un ; de délimitation).

IV. Arguments de debug

Activer l'affichage détaillé du traitement (-debug)

Le mode debug permet d'afficher le détail des traitements et de la représentation en mémoire du code. Cette option affiche notamment le résultat de la première structuration du code, et la liste des éléments récupérés (Fonctions, arguments, etc…).

Voici un exemple du résultat de cette fonction :

Activer cette option active également l'option –v du mode verbeux. Sous Windows, la taille de la console est modifiée pour faciliter la lecture.

Activer l'affichage détaillé des opérations (-debugOp)

Cette option permet d'afficher le détail du traitement des opérations des lignes de code. Attention, ce mode peut vite générer beaucoup d'informations. Le détail des éléments trouvé par l'interpréteur est indiqué (Variables naturelles, opérateurs, etc…), ainsi que la décomposition en mémoire des différentes opérations. Attention, la lecture doit se faire de bas en haut.

Voici un exemple de l'utilisation de l'argument :

```
[Info] >Debut traitement de la fonction start (ID 0)
[Info] >>Ligne de code a traiter :      a=2*(106+120)
[Info] >>>Var naturelle (2)
[Info] >>>Variable (a)
[Info] >>>Parentheses
[Info] >>>>>Var naturelle (120)
[Info] >>>>>Var naturelle (106)
[Info] >>>>>Operateur de calcul (+)
========== Liste des operations ===========
Operation #3 :  REF2+REF1
Operation #2 :  Valeur : 106
Operation #1 :  Valeur : 120
=========================================
[Info] >>>Operateur de calcul (*)
[Info] >>>Operateur d'affectation (=)
========== Liste des operations ===========
Operation #5 :  REF2=REF4
Operation #2 :  Variable a
Operation #4 :  REF1*REF3
Operation #1 :  Valeur : 2
Operation #3 :  (REF1)
Operation #1 :  Valeur : 120
```

Activer cette option active également l'option –v du mode verbeux ainsi que l'option -debug. Sous Windows, la taille de la console est modifiée pour faciliter la lecture.

V. Gestion des erreurs

Erreur du programme

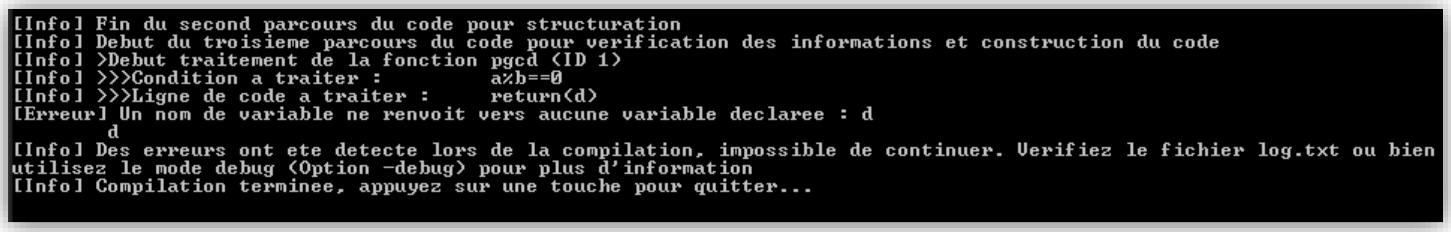
Il est possible que la compilation n'aboutisse pas si, par exemple, le programme n'arrive à correctement lire le fichier donné en entrée.

Le programme utilise également le fichier « **tmp** » pour récupérer les informations de compilation de **GCC** ou de **TinyCC**. Si un document porte ce nom à la racine du programme, celui-ci sera écrasé.

Erreur de code SPL

L'interpréteur vérifie que le code contenu dans le fichier à compiler est correct et respecte les règles du **SPL** (Voir document sur les règles du langage). En cas d'erreur dans le code, le programme arrête tout de suite la compilation et affiche l'erreur obtenue.

Il peut être utile d'utiliser le mode –debug pour situer le plus précisément l'erreur dans le code.



À noter que sous **Linux** les erreurs se colorent en rouge.

Erreur de compilation de TCC/GCC

En cas d'erreur dans la compilation de **TCC** ou de **GCC**, le programme affiche les raisons de cette erreur. Le code généré par le programme ne devrait pas poser des problèmes, mais d'autres facteurs peuvent empêcher de lancer la compilation.

VI. Fichiers nécessaires

Le programme nécessite le fichier **pthreadGC2.dll** sous **Windows**, et ne nécessite aucun fichier sous **Linux**.

Si vous souhaitez compiler le code généré par le programme, le dossier **SPL_bluitin** doit être dans le même dossier. Ce dossier contient l'équivalent de certaines fonctions « built-in » du **SPL** en **C**.