

TP de Compilation n° 1 : Prise en main de CerCo

06 Février 2014

Vous pouvez trouver tous les sujets au format électronique sur internet à l'adresse :

<http://www.pps.univ-paris-diderot.fr/~pboutill/CompilM1/>.

Ce sujet propose de vous familiariser avec le code source du projet-compilateur CerCo¹, pour lequel vous devrez améliorer certaines fonctionnalités à la fin du semestre. Dans un premier temps (environ la moitié du semestre), vous allez devoir rétablir certaines parties du compilateur qui ont été négligées. Cela vous permettra de comprendre des notions fondamentales à la compilation.

I) Récupération

Tout d'abord, vous devez récupérer le code source de CerCo. Tapez la commande suivante sous un shell :

```
git clone ~pboutill/git/CerCo.git répertoire /d /votre /choix/
```

Au cours du semestre, des modifications du code source pourraient être effectuées. Pour mettre à jour votre version du code, il suffira de taper la commande `git pull` sous un shell à partir de la racine de vos sources CerCo. Il est conseillé de le faire au début de chaque séance.

II) Compilation et exécution

1. La racine des sources contient un **Makefile**. Exécutez-le (tapez la commande `make` sous un shell) ; cela construira le programme `cc` qui se trouvera lui aussi à la racine : c'est le compilateur CerCo.
2. Vous pouvez alors connaître l'utilisation des options du compilateur en l'exécutant avec l'option `-help`. Tapez la commande `./cc -help` et observez le résultat.
3. Lorsqu'un fichier source C est passé en argument au compilateur CerCo, il est traduit en code assembleur MIPS et le résultat est sauvegardé dans le même répertoire que le fichier source. Le répertoire `tests/` à la racine des sources CerCo contient quelques programmes C. Essayez de compiler le programme `tests/rr_y_copy.c` à l'aide de CerCo. Vous verrez un erreur apparaître car la plupart des fonctionnalités du compilateur ont été désactivées.
4. Dans CerCo, la compilation d'un programme C en langage assembleur se fait en plusieurs *passes*, qui sont des successions de transformations dans des langages intermédiaires. CerCo permet de choisir à quel passage arrêter le processus de compilation avec l'option `-l` du compilateur. Pour l'instant, seul le premier passage de traduction, vers le langage Clight, est disponible. Tapez la commande `./cc -l Clight tests/rr_y_copy.c`. Le résultat se trouve dans le fichier `tests/01-rr_copy.c` ; comparez-le avec le code source original.
5. CerCo permet également d'inspecter ses résultats intermédiaires avec l'option `-i`. Par exemple, pour simuler l'exécution du programme `tests/rr_y_copy.c` au niveau du Clight, il suffit de taper la commande : `./cc -l Clight -i tests/rr_y_copy.c`. Essayez. Enfin, en ajoutant à la commande précédente l'option `-d`, vous verrez une *trace* d'exécution, qui peut faciliter la détection de bugs.

1. <http://cerco.cs.unibo.it/>

6. CerCo permet de compiler des programmes C *stand-alones*, c'est-à-dire qui n'incluent pas d'autres fichiers. En particulier, cela signifie que les fonctions des bibliothèques standards de C, comme `printf` de `stdio.h`, ne fonctionnent pas avec CerCo. Pour pallier à ce problème, CerCo reconnaît un certain nombre de primitifs d'entrée-sortie ; vous pouvez vérifier les en consultant l'interface du module `Primitive` dans le fichier `src/common/primitive.mli`. Le programme `tests/rry_copy.c` donne d'ailleurs un exemple d'utilisation de certains de ces primitifs.

III) Arborescence

Cette section présente une description rapide de l'arborescence des sources CerCo. Cela vous permettra de trouver rapidement les modules qui vous intéressent par la suite.

Chemin	Description
<code>compiler/</code>	Analyses lexicales et syntaxiques de C.
<code>src/</code>	CerCo à proprement parler, en particulier les modules principaux qui lient tous les autres. <i>Exemple</i> : le module <code>Langages</code> offre des fonctionnalités pour parser, interpréter, sauvegarder, etc.
<code>src/utilities/</code>	Modules utilitaires. <i>Exemple</i> : le module <code>StringTools</code> donne des outils sur le type <code>string</code> .
<code>src/common/</code>	Modules communs à plusieurs langages intermédiaires. <i>Exemple</i> : le module <code>Primitive</code> précédemment décrit.

Enfin, les répertoires `clight/`, `cminor/`, `RTL bs/`, `RTL/`, `ERTL/`, `LTL/`, `LIN/` et `ASM/` du répertoire `src/` concernent le langage respectif : interprétation, affichage, traduction vers le langage intermédiaire suivant.

I) Le module Dev_test

Le module `src/Dev_test` définit la fonction `do_dev_test` qui s'appelle avec l'option `-dev` de CerCo. Elle est utilisée pour que les développeurs puissent tester CerCo comme ils le souhaitent. Elle prend en argument la liste des fichiers qui ont été donnés sur la ligne de commande lors de l'appel à CerCo.

Modifiez la fonction `do_dev_test` du fichier `src/dev_test.ml` afin qu'il affiche le nom des fichiers passés en argument à CerCo (c'est-à-dire, affichez chaque élément de la liste `filenames`).

Une fois ceci fait, recompilez CerCo (`make` à la racine), puis testez le résultat en invoquant la commande `./cc -dev fichier1 fichier2 ...`

) CerCo complet

Une version complète de CerCo (sans les sources, juste l'exécutable) est disponible pour que vous puissiez comparer vos résultats tout au long du semestre : `~pboutill/cc`.