Greffons de GCC avec MELT *

Basile STARYNKÉVITCH
CEA, LIST (Saclay) [DILS/Labo Sûreté des Logiciels]
basile@starynkevitch.net
ou basile.starynkevitch@cea.fr
mobile: 06 8501 2359

5 février 2010

audience : développeurs seniors, chefs de projets logiciels importants [en C, C++, Ada, Fortran, compilés avec GCC], responsables de génie logiciel, méthodologie, outillage, qualité, tests ...

1 Introduction

La prochaine version (4.5) du **compilateur libre GCC** [1] paraît bientôt (2010T1) et fournit deux fonctionnalités majeures :

- *"link-time optimization"* optimisation à l'édition de liens : S'il est lancé avec options `-flto -O2` à la compilation *et* à l'éditions de liens de votre logiciel (application exécutable, bibliothèque), le compilateur effectuera des optimisations supplémentaires entre unité de compilation (et par exemple peut expanser les appels de fonction d'un fichier `f1.cc` en C++ vers `f2.c` en C ou `f3.f90` en Fortran).
- *"plugins"* greffons : on peut développer et utiliser des greffons qui étendent les nombreuses fonctionnalités de GCC pour *vos* logiciels. GCC devient ainsi adapté à votre code (diagnostics ou optimisations spécifiques). Par exemple `gcc -fplugin=melt.so -fplugin-arg-melt-mode=makegreen` va charger le méga-greffon MELT, l'utiliser dans le mode `makegreen` qui remplace les appels `fprintf(stdout,...)` par `printf(...)` - après "inlining" éventuel.

Les projets logiciels utilisant GCC peuvent dorénavant améliorer ce compilateur pour augmenter leur productivité.

2 Pourquoi des greffons dans GCC ?

Les greffons [2] permettent **l'extension du compilateur GCC pour vos besoins spécifiques**, qui devient ainsi *votre compilateur **sur mesure*** ! C'est le moyen d'ajouter dans GCC des fonctionnalités (propres à une entreprise, une communauté, un logiciel) telles que :

- *diagnostics spécifiques*, par exemple avertir de toute fonction qui ferait un appel à `fopen` sans en tester le résultat (nul si le fichier ne peut pas être ouvert).
- *optimisations spécifiques* (par exemple `fprintf(stdout,...)` → `printf(...)`) ou particulières à votre système (matériel/logiciel).
- "programmation par aspects", par exemple ajouter une impression avant chaque appel à votre fonction `foo` dont le premier argument est zéro.
- "*retro-ingénierie*" (refactoring), aide à la *navigation* dans le source de gros logiciels patrimoniaux (*"legacy code"*), *métriques* ...
- *validation de règles de codage* spécifiques à votre métier.
- tout autre chose !

L'extension (par un greffon) de GCC profite évidemment de toute la puissance du compilateur et de ses traitements et représentations internes.

Développer un greffon [3] n'a de sens que pour des logiciels importants (plusieurs centaines de milliers de lignes, ou certains développements particuliers) - nécessitant un effort d'outillage et de méthodologie- et requiert :

1. d'abord *la compréhension des spécificités de votre logiciel* et de votre métier ;
2. *un survol des représentations internes* essentielles de GCC (Gimple, Tree, ...) ;

*Les opinions énoncées ici sont seulement celles de l'auteur. `$Revision: 113 $`

1. `http://gcc.gnu.org/`, GNU compiler collection (licence GPLv3 ©FSF), pour C, C++, Ada, Fortran, Java ... vers plusieurs dizaines de machines cibles. GCC-4.5 comme MELT sont déjà disponibles en "snapshot".

2. Les greffons de GCC doivent être sous licence libre. Il faut les recompiler d'une version mineure à l'autre (4.5.0 → 4.5.1) et peut-être les adapter aux changements majeurs de version (4.5 → 4.6) du compilateur.

3. Ça demande un effort de plusieurs semaines ou mois (stage de fin d'études ou projet interne).

3. *une perception* sommaire de l'organisation (que votre greffon va améliorer) *des nombreuses* ($\approx$ 200) *passes de compilation* internes à GCC. Une commande telle que `gcc -O -fdump-tree-all t.c` génère des fichiers textuels (comme `t.c.024t.ssa` etc.) pour aider à comprendre les représentations internes après chaque passe, et donc à choisir la plus adaptée à vos besoins.

Le codage en C des greffons peut devenir fastidieux, mais MELT facilite ce codage (un fois comprises les représentations internes de GCC) par son formalisme. Au contraire d'autres outils externes (donc utilisés en dehors / en plus du cycle de développement habituel), parfois sophistiqués et puissants [4] ou onéreux, GCC, amélioré par *votre greffon*, s'utilise facilement sans trop perturber la chaîne de développement ou les habitudes des développeurs de votre logiciel.

3 l'outil MELT

MELT *"Middle End Lisp Translator"* [5] est lui-même un [méga/méta-] greffon de GCC qui offre un dialecte lispien spécifique pour **faciliter le codage des extensions** (greffons) de GCC, notamment :

- des facilités puissantes de filtrage *"pattern matching"* sur les représentations internes (quasi-arborescentes) de GCC (Gimple est commun à tous les langages sources et les systèmes cibles). On peut écrire très concisément en MELT des filtres sur toute représentation interne de GCC (et le filtrage est à la base de tout vos traitements internes à GCC).
- le traitement des données habituelles de GCC comme des valeurs de MELT [6].
- un style de programmation applicatif ou à objets soutenu par un gestionnaire automatique de la mémoire.
- une très forte intégration avec les internes de GCC, et la faculté, si besoin était, d'insérer un peu de code C dans du code MELT et d'invoquer souplement les fonctionalités internes de GCC.
- l'efficacité : le code MELT est traduit en du C (suivant le style de GCC), lui-même compilé en un module chargé dynamiquement à la volée.
- une syntaxe infixe sera bientôt disponible (contre l'allergie aux parenthèses)

MELT est déjà disponible comme branche (GPLv3 ©FSF) et sera diffusé comme greffon dès la parution de GCC 4.5.

Je suis disposé à vous aider à coder vous-même vos greffons en MELT, ou à vous former sur les greffons. N'hésitez pas à me contacter. Il y aura un exposé *Greffons de GCC* à SOLUTIONS LINUX (Paris, mi-mars 2010).

Un projet logiciel innovant a donc intérêt à étendre sur mesure son compilateur GCC avec MELT, obtenant ainsi un outillage propre à ses besoins.

English summary

The GCC free compiler (4.5) will provide two major features : link-time optimizations and plugins, which enable **customization of the GCC compiler to your own needs** (specific optimizations, diagnostics, refactoring or navigation tools, ...). Coding plugins in C is tedious, so the MELT extension provides you with a higher-level language to develop such specific GCC facilities. MELT gives you pattern-matching, applicative and object programming, ... to ease your particular GCC extensions suited to your own needs. GCC-4.5 and MELT are free (GPLv3) software, soon released, with already existing snapshots. Ask me for help !

4. Comme Frama-C, analyseur statique libre `http://frama-c.cea.fr` de code C, prouvant que les fonctions sont conformes à leurs spécifications en ACSL.

5. `http://gcc.gnu.org/wiki/MiddleEndLispTranslator`

6. MELT suit facilement l'évolution de GCC et sait traiter aussi bien des valeurs de premières classes (fonctions, objets, listes, tables, nombres enboîtés *"boxed"* ...) que les choses [données natives] de GCC- Gimple et Tree notamment. MELT fournit plusieurs mécanismes d'interfaçage au code de GCC.