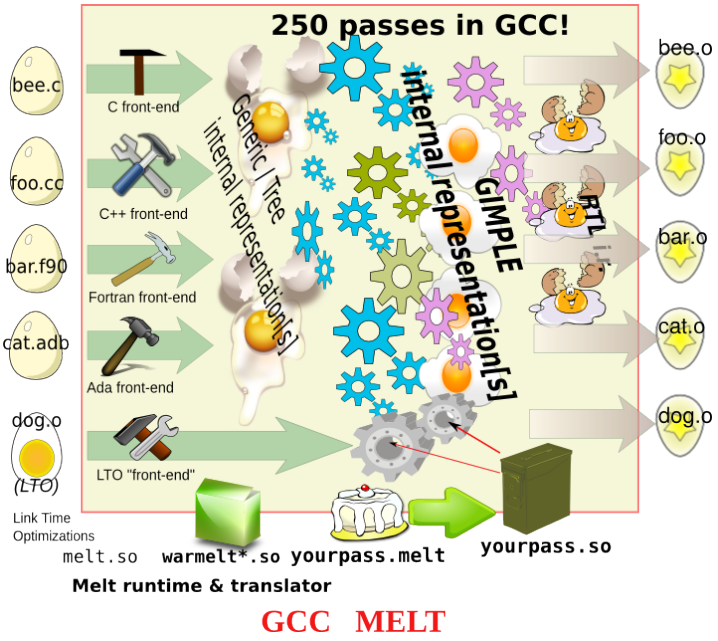
Customizing *your* GCC compiler with MELT*

Basile STARYNKÉVITCH
CEA, LIST (Saclay, France) [Labo Sûreté des Logiciels]
basile@starynkevitch.net or basile.starynkevitch@cea.fr
mobile: +33 6 8501 2359, office: +33 1 6908 6595

February 2, 2011



Audience: senior software developer or project leader for large or legacy software coded in C, or C++, Ada, Fortran, ... compiled by GCC; senior software architect; quality insurance, sub-contracting engineers; methodology or software engineering tools expert; large or strategic software projects.

Abstract

The free GCC[1] compiler can be extended with plugins. This permits extensions customizing GCC for a *specific* industry, corporation, or large software project (in C, C++, Ada, Fortran, ...), to provide particular diagnostics, optimizations, or custom tools working on your source code base.

Customizing GCC requires understanding its internal representations and passes. But coding your GCC extension as a plugin in C is not that easy (because C is not well suited to handle compiler's internal representations).

The MELT tool (GPLv3 free software) provides you with a domain specific language tailored to GCC internals. The MELT language has a familiar Lispy look, and gives you several features (functional, applicative, object-oriented, reflexive programming styles, powerful pattern matching, strong interface to GCC internals) to ease development of your specific GCC extensions.

1 Why do *you* need to customize GCC?

Any large, legacy, or business-strategic software base (for instance ≈ 1 million lines of C, C++, Fortran, Ada, ObjectiveC, ... source code compiled by GCC) has been developed for several years by a team of qualified software developers so represents a significant investment.

*Opinions are only mine, not those of my employer CEA or of the GCC community. Work funded by French DGCIS thru GlobalGCC [ITEA] and OpenGPU [FUI] projects. `$Revision: 160 $`

[1] GCC = the Gnu Compiler Collection, `http://gcc.gnu.org/` is the leading free [straight or cross-] compiler for C, C++, Fortran, Ada, Objective C (and Java), ..., on dozens of systems (Linux, AIX, Windows, ...) and processors (x86, AMD64, ARM, Sparc, PowerPC, MIPS, ...). `gcc-4.5.1` was released in august 2010.

Each such software has its own peculiarities and habits, specific to your project or system, your corporation, an industry, an application domain. So you can take profit of *your* **specific** GCC extensions[2] to get :

- additional ***warnings*** of misuse of some functions
  Example: warn if a call to `fopen` don't check that the resulting opened `FILE*` is not null.
- extra ***type checks***; Examples: check that a pointer argument is never null in every call of some function. Typing variary functions like `printf` or `g_object_set` in GTK.
- contextual ***optimizations***; Example: `fprintf(stdout,...)` ⇒ `printf(...)`
- ***coding rules*** validation; Examples: In C++, "ensure base classes common to more than one derived class are virtual" (HICPP 3.3.15). With pthreads, "Every call to `pthread_mutex_lock` should be followed by a similar call to `pthread_mutex_unlock` in the same block."
- ***source code navigation*** or metrics, or any other ***processing***, including ***aspect-oriented programming***, ***retro-engineering*** and ***re-factoring*** tasks. Whole program processing becomes possible (with *Link Time Optimizations* GCC ability).

Each above goal amounts to finding **patterns** in GCC internal representations of your source code, and you will need to partly understand GCC internals to realize it, and to *know very well your code* base and habits to specify them. Coding such extensions in C is impractical and not cost-effective (since GCC is a complex and growing 4.5MLOC software). Gluing scripting languages (like Python, Ruby, Ocaml ...) into GCC is not realistic.

2 What MELT can bring you?

MELT is an available free software (GPLv3) GCC plugin (or branch) to ease the development of your specific GCC extensions. It provides a ***domain specific language*** suited to GCC internals, to easily code your own GCC extensions in. Notable features include:

1. your MELT high-level code is translated to C source in the style of GCC internals.
2. the MELT language has a familiar and simple Lispy syntax (looks like Scheme, Common Lisp or Emacs Lisp).
3. The MELT translator (to C) is bootstrapped and implemented in MELT so exercises most of it.
4. The MELT run-time has an efficient generational garbage collector compatible with GCC internals.
5. MELT code can contains C chunks, and interfaces most GCC internals (Gimple, Tree, passes, ...).
6. MELT has very ***powerful pattern matching facilities*** suited to GCC internals.
7. ***high-level programming styles*** (functional, applicative, object-oriented, modules, reflective abilities, . . .)
8. handle both MELT values (objects, closures, boxes, . . .) and raw GCC stuff.
9. a documentation generator (from your future small piece of MELT code).

→ Use MELT to customize GCC for your *own* needs !

See **`http://gcc.gnu.org/wiki/MELT`** (now) or ***`http://www.gcc-melt.org/`*** (end of october 2010).

[2] For instance, the Mozilla project has developed their own GCC plugin, TREEHYDRA to help developers of the FIREFOX browser.