Gont Features

September 16, 2002

`$Rev: 997 $ $LastChangedDate: 2002-09-16 09:01:37 +0200 (Mon, 16 Sep 2002) $`

Contents

1 What's Gont?

Largely imperative composition of C-like lexical layer, control structures and speed with ML's typesystem, functions as first class citizens, and safety. Plus possibly few more things, like objects.

1.1 Safety

Gont is safe. This means that compiler shouldn't produce any code, execution of which would cause SEGV. Parse: it should not be possible to overflow buffers, mismatch types of arguments in calls and so on. However note, that SEGV can be obtained from playing with `extern` declaration, compiler has no means of validating.

1.2 Level of abstraction

Gont is very higher level language.

¡evangelisation¿

> [...] And, actually, the more you can avoid programming in C the more productive you will be.
>
> C is very efficient, and very sparing of your machine's resources. Unfortunately, C gets that efficiency by requiring you to do a lot of low-level management of resources (like memory) by hand. All that low-level code is complex and bug-prone, and will soak up huge amounts of your time on debugging. With today's machines as powerful as they are, this is usually a bad tradeoff – it's smarter to use a language that uses the machine's time less efficiently, but your time much *more* efficiently. Thus, Python.
>
> Eric S. Raymond

This is quotation from hacker howto by ESR :-). One may easily note that it does not only talk about Python.

¡/evangelisation¿

1.3 Memory Management

Gont provides transparent memory management, along with garbage collection (the particular kind of garbage collector used is Boehm conservative garbage collector, used also by GCJ and Popcorn). It generally means you can allocate as much as you want, forget the pointers, and the Gont runtime system with get rid of unused memory.

2 How Gont compares to...?

2.1 ML

How is Gont different from Caml or SML? (Caml and SML are functional languages, with imperative features from ML family) Hm... generally all languages are interchangeable, what can be written in one, can be written in all other. However in real life it is rather important how easy can you get the code to work, how much bugs will compiler detect (vs bugs left for the programmer) and how fast will it run. Gont places accents on these things somewhere between Caml and C. Generally it does not provide as much support for functional programming as Caml does, similar can be told about Gont's module system (which is a toy, compared to functors and other ML machinery) and restricted polimorphism. On the other hand, linking Gont code with C is very easy, the only thing you need to remember, is not to put pointers from Gont, in `malloc()`'ed area – save it on stack, or in `GC_malloc()`'ed area. Interfacing OCaml is... ghm... nightmare, mainly because of its precise garbage collector.

2.2 Popcorn

How is Gont different from Popcorn? (Popcorn is safe C subset, compiler is available to TALx86 (Typed Assembly Language)). Popcorn was inspiration for Gont :) However, it is somewhat limited (especially to x86) and not currently under development (AFAIK).

2.3 Cyclone

Cyclone (`http://www.cs.cornell.edu/projects/cyclone/`) is language similar to Gont in the same sense as C++ is similar to Java. It's a dialect of C designed to be safe: free of crashes, buffer overflows, format string attacks, and so on. Cyclone has more powerful typesystem – it includes regions, that allow not to use garbage collection all the time. It also has pointers in C's sense. OTOH Gont is not dialect of C. It does not try to be backward compatible. This results in much smaller language, probably easier to understand at first.

2.4 C++

Gont is not compatible with C. C++ tries to be. Popcorn, Cyclone and Java are all far more C-like then Gont is.

Gont currently does not have even the very limited amount of objective features C++ has. This should change in future. OTOH Gont polimorphic typesystem, with functions as first class citizens, is far more powerful then the one that can be found in C++.

2.5 Java

Gont in intention should be equally easy/hard to understand at first, as Java is, but be able to provide nice machinery, like patterns and polimorphism, later.

3 Compiler stuff

3.1 What's Ksi?

Ksi is intermediate language that Gont compiler outputs. Ksi looks like Lisp, but is rather close to C in spirit. Ksi is compiled by GCC front end.

3.2 Does Gont compiler reconstruct types?

Yes. However few operators are ad-hoc polymorphic, so there is often need for explicit type annotations.

Also it is allowed to have field named let's say `next` in two structures `foo` and `bar`. In this case there is no way to tell type of `x.next` if type of `x` is not yet known. Type annotation is requires also in this case.

Additionally functions are typechecked sequentially, and it is error to call global function that has no explicit types and has not been yet typechecked.

4 What do I need to run Gont compiler?

First of all, you need an operating system and platform, supported by GCC and Boehm GC. At the very moment Gont compiler is known to work under x86-linux and ppc-linux. alpha-linux testing is in progress.

If you managed to run it under some different OS/arch, please tell us.

Following software packages are needed to compile Gont:

- GCC, version 3.1.1 or later (3.2 is recommended). You need GCC sources, in order to compile Ksi. Ksi pre55 is required by current version of Gont.
- GNU Make (no special version requirements, I guess :)
- GNU Bison is required to build Gont parser. Reasonably new version is required, one that supports `%locations` feature. I use 1.35.
- Boehm Conservative Garbage Collector, version 6.0 or later I guess... Gont **will** build without Garbage Collector. However it won't free any allocated memory then ;) We have a local copy.
- Perl version 5+
- libxml version 2.4+
- pkgconfig is needed in order to determine how to use libxml2, so if you are compiling things on your own, make sure you first compile pkgconfig and then libxml2, so the later will detect the former and install appropriate `libxml-2.0.pc` file.
- Libc with `iconv()` function. This the case on linux with glibc2, need some info about other systems (I've seen -liconv somewhere... anyway `iconv()` is in my copy of SUSv2, so there should be not much problems on reasonable and recent systems)
- If you want to generate documentation, you will need Hevea, TEX (and LATEX). XSPP XSL preprocessor and libxslt are needed for libref generation. If you are not going to change documentation, you probably don't need it.
- usual set of rm/mv/cp/grep/sed etc.

All these packages (including Ksi compiler, but not Gont compiler, yet ;) are provided as RPM packages by PLD Linux Distribution, but some only by NESt PLD branch.

5 Administrativia

5.1 How to contact Gont developers?

There is mailing list available. To subscribe it, send empty email (subject also ain't important) to gont-subscribe@pld.org.pl. The list address is gont (at) pld org pl (of course, change (at) to @, and spaces to dots, this is to protect against spam, if you are reading pdf – sorry :¡). You can post to the list, even if you are not subscribed.

5.2 Who wrote Gont compiler?

Gont compiler is being written at the University of Wroclaw, Computer Science Institute, by Michal Moskal, Kamil Skalski and Marek Langiewicz. Gont compiler is web-hosted on team.pld.org.pl, and ftp-hosted on ep09.kernel.pl.