



Published on *ROOT* (<http://root.cern.ch/drupal>)

[Home](#) > [Blogs](#) > [axel's blog](#) > [Printer-friendly PDF](#)

## Do we need yet another custom C++ interpreter?

Submitted by axel on Thu, 22/12/2011 - 22:13

[cint](#) [1] [Cling](#) [2] [dictionaries](#) [3] [python](#) [4]

Hi,

"A ROOT User" [asks](#) [5] "Is it really necessary to replace CINT dictionary with cling?", bringing up very reasonable concerns and arguments against re-implementing CINT. I will try to answer his comments to clarify why we do it, and how it connects with the rest.

A fundamental misconception is that the status quo is acceptable. It is not, for several reasons.

1. *CINT vs C++*  
CINT was designed (20 years ago!) to be a C interpreter; C++ support was added later. It still has many shortcomings with C++ 2003, let alone C++11.
2. *CINT maintenance*  
The original author of CINT, Masaharu Goto, has moved on; CINT has been maintained mainly by the ROOT team. It has 300k lines of code; that's a considerable fraction of ROOT's 2.5MLOC. It has been designed to fit into an integrated processing unit of appliances (like medical ones) - not for 16GB RAM, 8 compute thread, 50000 class environments.
3. *Reflex and GCCXML solve it*  
ATLAS, CMS and LHCb use GCCXML to parse their headers, a set of python scripts to parse the generated XML file and write a C++ source file, the Reflex dictionary, which then gets compiled, linked, loaded, its data injected into the Reflex reflection database, which then gets copied through Cintex into CINT. We have thus many duplications of strings (three in the worst case with Reflex) and conflicts between duplicate dictionaries in Reflex versus CINT (famous: "std::map<std::string, TH1\*>" must *not* be described through Reflex). On top of that, GCCXML is a limited parser (e.g. it swallows typedefs in certain conditions, think Double32\_t); as it uses the GCC parser this will not be fixed. I.e. the current setup is fragile, inefficient, and limiting.
4. *CINT is not relevant, I use PyROOT*  
For calling into C++, PyROOT relies on CINT's reflection data from ROOT (which is why it's so fantastic compared to static SWIG-based approaches). And ROOT relies on CINT for I/O, both the dictionaries and the interpreter. I.e. you use CINT much, much more than you think: it's not just the prompt, it's in the core of most of ROOT.

So we [need](#) [6] to do *something*. C++ interpreters are [extremely rare](#) [7]. Instead of rewriting a C++ interpreter we decided to reuse existing code. Code that we can still influence, but that's nevertheless production-grade. We expected that this will solve the maintenance and correctness issues. And because it's correct we don't need Reflex, but can instead use one central, fast (compiler!) reflection database.

So yes, this is a major overhaul of ROOT and the dictionaries. We will signal that with a new major ROOT version number. But we expect it to solve the correctness, stability, memory and CPU-consumption as well as the maintenance issues we currently have. The current implementation of cling (which is not yet complete) uses a mere 5000 lines of custom code developed by HEP; everything else is provided through LLVM and clang.

And regarding PyROOT: I am sure Wim will make good use of the new JIT power that comes with cling! Just like we expect the JIT to leave traces e.g. in TFormula, and the real reflection database in the I/O, THtml etc. It gets us unstuck, flexible and future-safe in many central areas of ROOT. O the places you'll go!

Cheers,

**Source URL:** <http://root.cern.ch/drupal/content/do-we-need-yet-another-custom-c-interpreter>

**Links:**

- [1] <http://root.cern.ch/drupal/category/package-context/cint>
- [2] <http://root.cern.ch/drupal/category/package-context/cling>
- [3] <http://root.cern.ch/drupal/category/package-context/dictionaries>
- [4] <http://root.cern.ch/drupal/category/package-context/python>
- [5] <http://root.cern.ch/drupal/content/dictionaries-cint-and-cling#comment-856>
- [6] <http://root.cern.ch/drupal/content/requested-cling-features>
- [7] <http://www.google.com/search?aq=f&sourceid=chrome&ie=UTF-8&q=%22C%2B%2B+interpreter%22>