



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

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

## Dictionaries in CINT and cling

Submitted by axel on Wed, 07/09/2011 - 11:38

[cint](#) <sup>[1]</sup> [Cling](#) <sup>[2]</sup> [dictionary](#) <sup>[3]</sup>

Hi,

Marcelo [asked](#) <sup>[4]</sup> about how I see the future of dictionaries with cling, if we manage to replace CINT with cling. Given that many people probably don't know what those "dictionaries" really do, I decided to post it! I'll keep it as simple and short as possible.

CINT sees code that you enter as strings, e.g. `TGraph g(12);` is really `"TGraph g(12);"`. It analyzes the input, sees that a variable `g` of type `TGraph` is declared. Then there is a constructor to be called, the one taking an `int`. Now, several things need to happen:

- Overload resolution: which constructors are there, which one should be invoked for an argument list `(int)`? CINT stores the available functions in the dictionary, with their names and argument and return types as strings.
- CINT needs to be able to call that constructor. Those are the `G__cpp...` functions ("wrappers" or "stubs") in the dictionary: they all have the same signature, and convert from a `void*` array to the function's actual parameter types.
- The argument is `"12"`, that needs to be converted into an integer value of 12 so it can be passed to the constructor.

ROOT I/O needs that, too: when reading an object's data from a ROOT file, it creates the object in memory (calling the default constructor) and then sets the data members to the values read from disk. To not do the complex steps above, ROOT stores a shortcut in the dictionary. There are more shortcuts in the dictionary, e.g. to tell ROOT how to create the `TClass` object for a class. And the dictionary implements functions that are declared by using `ClassDef()`, e.g. `Streamer()`, `ShowMembers()`. `ClassImp()` is basically unused nowadays, it's just for `THtml` to find the source file.

Part of cling is a compiler, clang. Of course for a compiler, `#include "TObject.h"` has all the information to describe the class. It is as good as a dictionary for `TObject`. What's missing are suppressions of classes, i.e. classes that should not be used for I/O or within the interpreter - but that we can add independently. And because clang uses LLVM and its just-in-time compiler we can call functions without stubs from CINT's dictionary.

The only missing part are the implementations of the `ClassDef()` functions. We don't need to generate them as source anymore; if we really need them we can just-in-time compile them. But we believe we can just get rid of them.

As you can see there is nothing left from the dictionaries: a simple `#include "TObject.h"` (or its precompiled header) is enough for cling! So let's cross our fingers that Vassil can stay with us and that we can pull it off: cling and its integration into ROOT!

Cheers,

© 1995-2013 The ROOT Team

Source URL: <http://root.cern.ch/drupal/content/dictionaries-cint-and-cling>

### 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/dictionary>
- [4] <http://root.cern.ch/drupal/content/cling-goes-public#comment-805>