ROOT6 and Backward Compatibility

Hi everyone, dear Matt!

Matt Walker has posted an extensive review [1] of ROOT and what he would hope the future of ROOT to be. Because I think many of his comments are good ones, and because I have heard some of them from several people in the past, I decided to give the answer to an audience that's little bit wider, in a dedicated post.

Backward Compatibility with CINT

We will discontinue support for using '.' and '->' interchangeably. Note that references have the identical "performance issue" as pointer derefs and use '.' and that '.' often results in a memory load, too, so I don't take performance differences as an argument :-) But we want to encourage proper C++. We do provide most of the interpreter extensions, but they can be turned off.

Class Hierarchy

A pet peeve of many, where it's not even clear how much your average novice physicist really cares. None of the real - in contrast to "I'd rather want to do computing"-physicists like me :-) - physicists I talked to ever saw that as an issue. That said: there is a Jira task [2] assigned to it; Lorenzo will investigate it. If you have suggestions don't hesitate to comment on the Jira ticket!

Templates

ROOT was limited in the use of templates due to the way CINT called functions. Now that we have a just in time compiler we will be able to instantiate templates at runtime, and call their functions as needed. That's e.g. the main reason why the TTreeReader [3] only really works well in ROOT6. We will migrate interfaces where it's useful, but due to the lack of documentability of possible template arguments (where concepts would help), novices usually find templates repelling compared to traditional classes.

C++11

There are several parts to it: ROOT needs to be able to parse C++11 code, and ROOT 6 will be able to do that out of the box when built with C++11 turned in. The next part is the accessibility of C++11 features through its reflection interfaces (e.g. TClass) - that will be done after ROOT 6. And the last part of C++11 support is adding features to ROOT's I/O where needed. O and by the way: we'd love to use C++11 also in the ROOT interfaces (where it makes them more readable / compact / performant), but we have to wait until *all* experiments have migrated as you cannot mix and match C++ 2003 and 11. And yes, I agree that C++11 is a new language :-)

FFT

Please open Jiras tickets to discuss design issues. I would have copied and pasted your comments into one, but you might want to reformat / restructure and explain some parts of your comments on that, at least I didn't want to preempt that.

Re-implementing ROOT

You won't believe how often we hit a situation where we wish we could make existing interfaces non-existent or unused. But as soon as we have users of code we try to *not* change the interfaces anymore, unless we have a very good reason for it. Otherwise the experiments' update to new ROOT versions would eat up lots of manpower - and their job is physics, not beauty of coding. Physics doesn't change a bit by adding templates. We don't want to have 20000 angry physicists (minus Matt) in our corridor, protesting because we broke all their code ("but your new code will be much nicer!"). That would be an epic failure [4] in addressing customer needs.

So what we need is an alternative program, something that comes after ROOT. Something that challenges ROOT - not by a wish list, but by an alternative implementation that's powerful and draws users. Until then: please do keep your comments flowing; we hear them and try to address them!

Cheers,

Source URL: http://root.cern.ch/drupal/content/root6-and-backward-compatibility

Links:
[1] http://root.cern.ch/drupal/content/do-we-need-yet-another-custom-c-interpreter#comment-1028
[2] https://sft.its.cern.ch/jira/browse/ROOT-5062
[3] https://sft.its.cern.ch/jira/browse/ROOT-5165
[4] http://techland.time.com/2013/05/06/microsofts-strategic-blunder-with-windows-8/