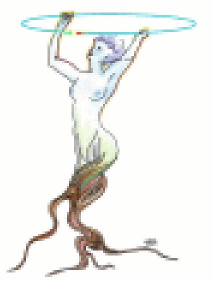


ROOT C++ Coding Conventions

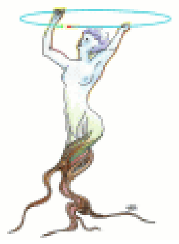
Fons Rademakers

The Taligent Coding Conventions

- When we started with ROOT in early 1995 the people with the largest body of C++ code were Taligent (an Apple/IBM/HP consortium writing a new OO C++ based OS)
- They had published their coding conventions and programming practices in a small book:

  Taligent's Guide to Designing Programs

The ROOT Coding Conventions



- Being new in C++ and with not much time to investigate the subject we decided to just adopt the Taligent conventions
- After a few months of experience we modified a few rules to our personal liking:

  The ROOT Coding Conventions

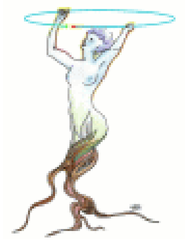
Naming Conventions

- Full adoption of the Taligent conventions
- ROOT changes and additions:
  - Append `_t` to typedefs and simple structs, e.g.:
    - `typedef int Int_t`
    - `struct simple_t { … };`
  - Don’t use M or V for mixin and virtual classes

Class Definition Order

- In ROOT we decided to use:
  - Friend declarations
  - Private members
  - Private methods
  - Protected members
  - Protected methods
  - Public methods
- Never spread member declarations around
- This directly shows the developer the meat of the class

Header (.h) File Layout



- CVS identification line
- Author statement
- Copyright statement
- Multiple inclusion protection
- Class description comment
- Protected includes
- Class definition
- Example header file

Source (.cxx) File Layout

- CVS identification line
- Author statement
- Copyright statement
- Class description comment
- Include statements
- Class static definitions
- Method implementation
- Example source file

Using Comments to Document the Code

- Data member description comments
- Class description comments
- Member function description comments
- Embedded HTML in comments

```
//begin_html
/*
<img src="gif/hsum.gif">
*/
//end_html
//begin_html <img src="gif/hsum.gif"> end_html
```

- See a raw header file and the hyperized version

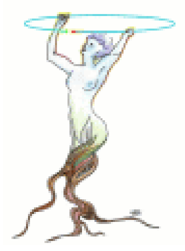
Preferred Coding Style

- See the Conventions page

Namespaces

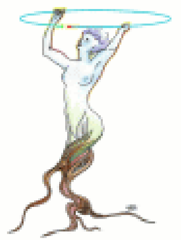
- Global functions are in the ROOT namespace
- Not yet any sub-namespaces, do we need those? Just ROOT namespace and class name convention should be enough
- To many levels of namespaces and class name reuse obfuscates the code

The LCG C++ Coding Guidelines



- Except for the specific Taligent/ROOT naming conventions and class definition order the LCG Coding Guidelines are basically the same as the ROOT Coding Conventions

Conclusions



- Coding conventions for large bodies of code that have to be maintained over a long period of time by many different people, often not the authors, is essential
- Adhering to the conventions is mandatory, the slightest deviation will confuse the readers and make them waste precious time