cetlib - Support #11826

Policy on suppressing known warnings

02/24/2016 03:58 PM - Ben Morgan

Status:	Closed	**Start date:**	02/24/2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	1.50 hour

Description

In compiling cetlib with native Clang on Mac (plus Boost 1.58 compiled against C++14), I've found an error when compiling cetlib sources and tests with the diagnostic level set to CAUTIOUS or higher.

- Boost's headers will emit a high number of warnings - can be suppressed by adding the include dirs as SYSTEM type (so -isystem is used in place of -I).
- extern "C" blocks fail with the "error":

```
  test/1_1_1.cc:6:15: error:
      'idString' has C-linkage specified, but returns user-defined type
      'std::string' (aka 'basic_string<char, char_traits<char>, allocator<char>
      >') which is incompatible with C [-Werror,-Wreturn-type-c-linkage]
  std::string idString() { return "1_1_1"; }
                ^
1 error generated.
```

These are not strict errors (the Boost warning are in Boost itself, the C-linkage is to give demangled names for plugin loading), so I was wondering if CET have a policy/recommended technique for suppressing known but benign or spurious warnings?

I can think of using pragmas, adding/removing flags for certain source files or system includes as needed, but I couldn't see anything obvious in cetbuildtools or upstream projects. In that sense it is perhaps more a question for cetbuildtools, but I wanted to give a concrete example of a use case (o.k., using native compiler...).

History

#1 - 02/24/2016 04:57 PM - Christopher Green

- Status changed from New to Feedback

We actually just had an example of this updating to Boost 1.60.0, which will feature in release 1.19 of the art suite. Our convention, and advice when asked, is to use pragmas and push/pop, which I **think** are also honored by clang?

See cetlib:source:cetlib/quiet_unit_test.hpp for an example.

We then encourage people to use the "quiet" header rather than repeating the pragma mantra everywhere. If the problems with Boost are more systemic under clang, then perhaps something like -isystem would indeed be warranted -- that would be a simple matter of adding the SYSTEM option to the include_directories invocation in cetbuildtools:source:Modules/FindUpsBoost.cmake@411bb10b#L91.

Does that answer your question?

#2 - 02/25/2016 02:02 PM - Ben Morgan

Christopher Green wrote:

> We actually just had an example of this updating to Boost 1.60.0, which will feature in release 1.19 of the art suite. Our convention, and advice when asked, is to use pragmas and push/pop, which I **think** are also honored by clang?
>
> See cetlib:source:cetlib/quiet_unit_test.hpp for an example.
>
> We then encourage people to use the "quiet" header rather than repeating the pragma mantra everywhere. If the problems with Boost are more systemic under clang, then perhaps something like -isystem would indeed be warranted -- that would be a simple matter of adding the SYSTEM option to the include_directories invocation in cetbuildtools:source:Modules/FindUpsBoost.cmake@411bb10b#L91.
>
> Does that answer your question?

Yep, that's great, thanks!

I'll look at the quiet header and otherwise use pragmas if necessary (that's what I've done before with Boost/clang or old Boost/new GCC).

Edit: Looks like I don't have perms to close the issue myself, but it is resolved/closed!

#3 - 02/29/2016 11:44 AM - Kyle Knoepfel

- Status changed from Feedback to Closed

- % Done changed from 0 to 100

We will be adjusting find_ups_boost to use the SYSTEM option with include_directories so that current code will not need to be adjusted.

#4 - 02/29/2016 11:51 AM - Lynn Garren

This change is in cetbuildtools v4_18_04.