QGIS Application - Feature request #234
ISO C++ forbids casting between pointer-to-function and pointer-to-object

2006-08-12 01:28 AM - Mateusz Loskot -

Status:	Closed		
Priority:	Low		
Assignee:	Gary Sherman		
Category:	Build/Install		
Pull Request or Patch supplied:		**Resolution:**	wontfix
Easy fix?:	No	**Copied to github as #:**	10293

Description

In file *src/core/gsproviderregistry.cpp* compiler reports following interesting warnings:

```
qgsproviderregistry.cpp: In constructor 'QgsProviderRegistry::QgsProviderRegistry(QString)':
qgsproviderregistry.cpp:116: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
qgsproviderregistry.cpp:124: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
qgsproviderregistry.cpp:125: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
qgsproviderregistry.cpp:137: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
qgsproviderregistry.cpp:146: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
```

In fact, casting issue reported by compiler should be considered as a bug.
This casting may cause undefined behaviour, it is not compliant with standard C++ thus it's considered as not portable construction.

Possible workaround but not solution is to use reinterpret_cast or std::memcp function to copy raw pointer data between pointer-to-function and pointer-to-data.

Here is a very interesting thread about this issue with where are also presented solutions I mention above and very detailed explanation of *why this is a bug* or *what problems may be expected* and *how to solve it as best as possible*:
casting to function pointer by Ulrich Eckhardt

History

#1 - 2006-08-12 01:29 AM - Mateusz Loskot -

I noticed this problem when building **Lib_Refactoring-branch** branch.

BTW, is this possible to add branches selection to Ticket Properties?

#2 - 2006-10-14 02:17 AM - Mateusz Loskot -

I'd like to point one more resource with explanation of this problem:

When standards collide: the problem with dlsym

This issue is also a subject of C++ Standard Core Language Defects:

195. Converting between function and object pointers

Finally, I'd suggest to *close* this bug with appropriate comment or mark as **enhancement** for future, when C++ Standard and implementations (compilers/libraries) are fixed.

Fortunately, it's not a bug in QGIS itself.

#3 - 2006-10-14 04:10 AM - Mateusz Loskot -

Changed by anonymous?!?

Please, anyone, **log-in if are attempting to change status of tickets**.
Otherwise it's unknown who to talk to about a ticket, if needed.

#4 - 2006-10-14 03:12 PM - Gavin Macaulay -

I'd suggest we mark it as an enhancement, and set the milestone to blank. That way it hangs around but doesn't show up in the 'tickets left in milestone' totals.

#5 - 2006-10-14 03:22 PM - Mateusz Loskot -

Perfect. I agree.
I believe the state should be changed by assignee.

#6 - 2006-10-14 05:12 PM - Gavin Macaulay -

The assignee is the default person for Build/Install tickets, so Gary probably hasn't explicitly asked for this one :)

#7 - 2006-12-18 04:07 AM - anonymous -

Actually, not even reinterpret_cast will work without warnings. One way to get around this (the behaviour is still undefined, strictly speaking, but at least the warnings go away) is to define a template function like this:

```
template <typename TO, typename FROM> TO nasty_cast(FROM f) {
  union {
    FROM f;
    TO t;
  } u;
  u.f = f;
  return u.t;
}
```

and then call it like this:

```
      void* foo = something();
        my_func_ptr = nasty_cast&lt;MyFuncPtrType&gt;(foo);
```

#8 - 2007-02-23 04:37 PM - Tim Sutton

Head builds without warnings now - we need to check if this report can be closed now. I reset the milestone to 0.9 since from now on we build with -Wall -Werror so if it exists in trunk it should be treated as a bug

#9 - 2007-07-21 09:25 PM - Tim Sutton

Changed to minor under the following scheme:

- blocker - bugs that should block the release. Since we are going to release pretty much 'come what may' I would like no bugs
- allocated to this category without consultation with me and / or PSC
- critical - bugs that cause the application to crash or corrupt data
- major - application features that do not function at all
- **minor** - features that function but imerfectly e.g. labels placing incorrectly
- trivial - gui useability issues or small issues with the documentation, install notes etc.

#10 - 2008-08-20 09:32 PM - Mateusz Loskot -

Tim,

Current version of QGIS SVN trunk does not set any flags like -Wall, -Werroretc. I have to do it manually:

```
cmake ... -D CMAKE_CXX_FLAGS:STRING="-pedantic -Wall -Wno-long-long -fstrict-aliasing -Wstrict-aliasing=1" ..
```

-Werror flag removed, becuase the baby does not build with it, warnings occur.

@anonymous

Agreed, because nasty_cast does the same what std::memcpy can do.

#11 - 2008-08-20 09:39 PM - Mateusz Loskot -

Tim,

Continuing my previous comment, current SVN trunk does still thrown the same warning if compiled with my set of CXX flags:

```
trunk/qgis/src/core/qgsproviderregistry.cpp: In constructor 'QgsProviderRegistry::QgsProviderRegistry(QString)':
trunk/qgis/src/core/qgsproviderregistry.cpp:115: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp:119: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp:127: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp:128: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp:138: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp:146: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
trunk/qgis/src/core/qgsproviderregistry.cpp: In member function 'QgsDataProvider* [[QgsProviderRegistry]]::getProvider(const QString&, const QString&)':
trunk/qgis/src/core/qgsproviderregistry.cpp:353: warning: ISO C++ forbids casting between pointer-to-function and pointer-to-object
```

#12 - 2008-08-23 07:47 AM - Mateusz Loskot -

Folks, as I see dlsym is used indirectly but through QLibrary::resolve. The http://doc.trolltech.com/4.0/qlibrary.html#resolve Qt documentation] says:

> *The symbol must be exported as a C function from the library.*

Does it happen in QGIS? In fact, it would solve the problem I've reported here.

Union trick, memcpy and other hacks do not, as C++ FAQ says: *It's illegal, period.*

#13 - 2008-08-23 09:10 AM - Jürgen Fischer

- Status changed from Open to Closed

- Resolution set to wontfix

Replying to [comment:13 mloskot]:

> *Folks, as I see dlsym is used indirectly but through QLibrary::resolve. The http://doc.trolltech.com/4.0/qlibrary.html#resolve Qt documentation] says:*
>
> *The symbol must be exported as a C function from the library.*
>
> Does it happen in QGIS?

Sure, otherwise resolve() wouldn't find the function. That's what QGISEXPORT is for.

> *In fact, it would solve the problem I've reported here.*

No, it wouldn't. The root of the problem is not QGIS, it's QLibrary::resolve() and in turn dlsym() (and maybe [[GetProcAddress]] on Windows, too). Those return void * and that result is impossible to legally cast to a function pointer in C++. Nevertheless that cast works fine, otherwise dlsym() would be useless.

If they'd return void (*)() there wouldn't be a problem at all. So anyone doing dynamic loading with C++ should have that problem at some point.

Although there might be architectures that use different memory and function pointers on which such a cast would be a problem (maybe there are embedded systems like that; using different memory for data and code), I doubt that those are a possible target for Qt. And I doubt that one would find dlsym() there.

The warnings just tell us, that our code wouldn't work there. And the tricks just make that warning go away. The point of it: compile in maximum warning level, get all the good warnings and not that one warning, that we believe doesn't apply to any architecture we possibly target.

And rereading the comments above - which I should have done in the first place - that was also your conclusion on 10/14/06. I think there's nothing we can do about it then simply wait. I take the liberty to close this bug as the problem will solve itself at some point, when we port to a Qt version, that supports the dlsym(), that does it right.

#14 - 2008-08-23 09:33 AM - Mateusz Loskot -

Jef,

I agree about where is the root of the problem and *extern "C"* does not solve all aspects of it. Clean and 100% valid solution for this problem does not exist. So, silencing warning does not eliminate it, potential risk of problems on various C++ implementations might be expected.

#15 - 2009-08-22 12:57 AM - Anonymous

Milestone Version 1.0.0 deleted

#16 - 2009-11-14 01:41 PM - Mateusz Loskot -

Warming up an old steak, I've learned that the union-based cast used in cast_to_fptr is not necessary. GCC accepts reinterpret_cast for purposes like in dlsym calls as an extension:

```
#ifdef +GNUC+
+extension+
#endif
isprovider_t *hasType = reinterpret_cast<isprovider_t*>(myLib->resolve("type"));
```