Exiv2 - Bug #1065

Is fileProtocol() thread-safe?

25 Apr 2015 18:03 - Max Pozdeev

Status:	Closed	**Start date:**	25 Apr 2015
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	basicio	**Estimated time:**	0.00 hour
Target version:	0.25		

Description

I use libexiv2 in multi-threaded app (2 parallel threads) and sometimes get a crash on function fileProtocol(). I see in debugger that both threads are in this proc and trying to fill the protDict var. Here is the trace.

First thread:

```
* thread #15: tid = 0xddef48, 0x00007fff8efe128e libstdc++.6.dylib`std::string::compare(std::strin
g const&) const + 10, name = 'DataCacherThread'
    frame #0: 0x00007fff8efe128e libstdc++.6.dylib`std::string::compare(std::string const&) const
+ 10
    frame #1: 0x000000001007ad6ed libexiv2.dylib`bool std::operator<<char, std::char_traits<char>,
std::allocator<char> >(__lhs=0x0000000100adb558, __rhs=0x0000000108103498) + 29 at basic_string.h:
2228
    frame #2: 0x000000001007ad5c1 libexiv2.dylib`std::less<std::string>::operator(this=0x0000000100
adb530, __x=0x0000000100adb558, __y=0x0000000108103498)(std::string const&, std::string const&) co
nst + 33 at stl_function.h:227
    frame #3: 0x0000000010080ff0d libexiv2.dylib`std::_Rb_tree<std::string, std::pair<std::string c
onst, Exiv2::Protocol>, std::_Select1st<std::pair<std::string const, Exiv2::Protocol> >, std::less
<std::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::_M_insert_unique(
this=0x0000000100adb530, __position=std::_Rb_tree<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::pair<const std::basic_string<char, std::char_traits<char>, std::alloc
ator<char> >, Exiv2::Protocol>, std::_Select1st<std::pair<const std::basic_string<char, std::char_
traits<char>, std::allocator<char> >, Exiv2::Protocol> >, std::less<std::basic_string<char, std::c
har_traits<char>, std::allocator<char> > >, std::allocator<std::pair<const std::basic_string<char,
 std::char_traits<char>, std::allocator<char> >, Exiv2::Protocol> > >::iterator at 0x0000000108103
3e8, __v=0x0000000108103498) + 493 at stl_tree.h:1019
    frame #4: 0x0000000010080fc5d libexiv2.dylib`std::map<std::string, Exiv2::Protocol, std::less<s
td::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::insert(this=0x00000
00100adb530, __position=std::map<std::basic_string<char, std::char_traits<char>, std::allocator<ch
ar> >, Exiv2::Protocol, std::less<std::basic_string<char, std::char_traits<char>, std::allocator<c
har> > >, std::allocator<std::pair<const std::basic_string<char, std::char_traits<char>, std::allo
cator<char> >, Exiv2::Protocol> > >::iterator at 0x0000000108103428, __x=0x0000000108103498) + 45
at stl_map.h:427
    frame #5: 0x0000000010080f097 libexiv2.dylib`std::map<std::string, Exiv2::Protocol, std::less<s
td::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::operator[](this=0x0
000000100adb530, __k=0x00000001081035a8) + 199 at stl_map.h:350
  * frame #6: 0x0000000010080d84d libexiv2.dylib`Exiv2::fileProtocol(path=0x0000000108103868) + 989
 at futils.cpp:252
    frame #7: 0x0000000010081831d libexiv2.dylib`Exiv2::ImageFactory::createIo(path=0x0000000108103
868, useCurl=false) + 45 at image.cpp:425
    frame #8: 0x0000000100818650 libexiv2.dylib`Exiv2::ImageFactory::open(path=0x0000000108103868,
 useCurl=false) + 64 at image.cpp:473

frame #6: 0x0000000010080d84d libexiv2.dylib`Exiv2::fileProtocol(path=0x0000000108103868) + 989 at
futils.cpp:252
   249                  protDict["ssh://"  ]   = pSsh;
   250                  protDict["file://" ]   = pFileUri;
   251                  protDict["data:"   ]   = pDataUri;
-> 252                  protDict["-"       ]   = pStdin;
   253              }
   254              for (Exiv2::protDict_i it = protDict.begin(); it != protDict.end(); it++) {
   255                  if (path.find(it->first) == 0)
```

Second thread:

```
* thread #16: tid = 0xddef49, 0x00007fff8efe128e libstdc++.6.dylib`std::string::compare(std::strin
g const&) const + 10, name = 'DataCacherThread', stop reason = EXC_BAD_ACCESS (code=1, address=0xf
fffffffffffff2)
    frame #0: 0x00007fff8efe128e libstdc++.6.dylib`std::string::compare(std::string const&) const
+ 10
    frame #1: 0x000000001007ad6ed libexiv2.dylib`bool std::operator<<char, std::char_traits<char>,
std::allocator<char> >(__lhs=0x0000000100adb558, __rhs=0x000000010ad09498) + 29 at basic_string.h:
2228
  * frame #2: 0x000000001007ad5c1 libexiv2.dylib`std::less<std::string>::operator(this=0x0000000100
adb530, __x=0x0000000100adb558, __y=0x000000010ad09498)(std::string const&, std::string const&) co
nst + 33 at stl_function.h:227
    frame #3: 0x000000010080ff0d libexiv2.dylib`std::_Rb_tree<std::string, std::pair<std::string c
onst, Exiv2::Protocol>, std::_Select1st<std::pair<std::string const, Exiv2::Protocol> >, std::less
<std::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::_M_insert_unique(
this=0x0000000100adb530, __position=std::_Rb_tree<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::pair<const std::basic_string<char, std::char_traits<char>, std::alloc
ator<char> >, Exiv2::Protocol>, std::_Select1st<std::pair<const std::basic_string<char, std::char_
traits<char>, std::allocator<char> >, Exiv2::Protocol> >, std::less<std::basic_string<char, std::c
har_traits<char>, std::allocator<char> > >, std::allocator<std::pair<const std::basic_string<char,
 std::char_traits<char>, std::allocator<char> >, Exiv2::Protocol> > >::iterator at 0x000000010ad09
3e8, __v=0x000000010ad09498) + 493 at stl_tree.h:1019
    frame #4: 0x000000010080fc5d libexiv2.dylib`std::map<std::string, Exiv2::Protocol, std::less<s
td::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::insert(this=0x00000
00100adb530, __position=std::map<std::basic_string<char, std::char_traits<char>, std::allocator<ch
ar> >, Exiv2::Protocol, std::less<std::basic_string<char, std::char_traits<char>, std::allocator<c
har> > >, std::allocator<std::pair<const std::basic_string<char, std::char_traits<char>, std::allo
cator<char> >, Exiv2::Protocol> > >::iterator at 0x000000010ad09428, __x=0x000000010ad09498) + 45
at stl_map.h:427
    frame #5: 0x000000010080f097 libexiv2.dylib`std::map<std::string, Exiv2::Protocol, std::less<s
td::string>, std::allocator<std::pair<std::string const, Exiv2::Protocol> > >::operator[](this=0x0
000000100adb530, __k=0x000000010ad095a8) + 199 at stl_map.h:350
    frame #6: 0x000000010080d84d libexiv2.dylib`Exiv2::fileProtocol(path=0x000000010ad09868) + 989
 at futils.cpp:252
    frame #7: 0x000000010081831d libexiv2.dylib`Exiv2::ImageFactory::createIo(path=0x000000010ad09
868, useCurl=false) + 45 at image.cpp:425
    frame #8: 0x0000000100818650 libexiv2.dylib`Exiv2::ImageFactory::open(path=0x000000010ad09868,
 useCurl=false) + 64 at image.cpp:473

frame #6: 0x000000010080d84d libexiv2.dylib`Exiv2::fileProtocol(path=0x000000010ad09868) + 989 at
futils.cpp:252
   249                  protDict["ssh://"  ]   = pSsh;
   250                  protDict["file://" ]   = pFileUri;
   251                  protDict["data:"   ]   = pDataUri;
-> 252                  protDict["-"       ]   = pStdin;
   253              }
   254              for (Exiv2::protDict_i it = protDict.begin(); it != protDict.end(); it++) {
   255                  if (path.find(it->first) == 0)
```

Related issues:

Related to Exiv2 - Bug #1066: Unable to build for Mac OSX 10.6	**Closed**	**25 Apr 2015**

Associated revisions

Revision 3729 - 25 Apr 2015 19:04 - Robin Mills

#1065. Fixing thread safety in fileProtocol. Thank You, Max for reporting this.

Revision 3730 - 25 Apr 2015 19:47 - Robin Mills

#1065. Thanks to Thomas B for spotting my error in omitting support for https.

Revision 3732 - 25 Apr 2015 20:43 - Robin Mills

#1065. std::map<int,const char*> doesn't build on MacOS-X 10.6 (Snow Leopard). Thanks Max for letting me know about this.

History

#1 - 25 Apr 2015 18:35 - Robin Mills

- Subject changed from Does fileProtocol() thread-safe? to Is fileProtocol() thread-safe?

- Category set to basicio

- Status changed from New to Assigned

- Assignee set to Robin Mills

- Target version set to 0.25

Max

You are right. It is not thread safe. It builds a little static map when it's run for the first time. If two threads hit it while it's being built - bad things can happen. I'll have to think about how best to have the dictionary built. We have tried to avoid having library initialization code. I'd also like to see if there are other little threading bombs hiding nearby!

My suggested work around for now is to call fileProtocol("") before you start spinning up threads and you'll probably be good.

#2 - 25 Apr 2015 19:11 - Robin Mills

Fix submitted. r3729. Thank you, Max for reporting this.

I've eliminated the (horrible) static and replaced it with a constant compiled-in array of static data. No need to initialize anything. I've reviewed the other two statics in futils.cpp and they are true static strings used by hex and base64 code.

Max:
I'm very pleased that you have a multi-threaded application. I won't mark this issue as "Resolved" at the moment. If/when you encounter additional thread-safety issues, please update this issue report and I will give it attention.

#3 - 25 Apr 2015 19:38 - Thomas Beutlich

Any reason why pHttps was removed by r3729 from the map?

#4 - 25 Apr 2015 19:48 - Robin Mills

Yes, there's a reason. I'm dyslexic. I counted the protocols in the Protocols enum in futils.hpp and decided everything was OK. Thank Goodness you're awake Thomas. Well spotted. Fix submitted r3730.

#5 - 25 Apr 2015 19:58 - Max Pozdeev

Thank you for quick response. I'll do more tests, I want to use multi-threading in production.

#6 - 25 Apr 2015 20:19 - Max Pozdeev

Just for info. I can not build latest revision (r3730) on Mac OSX 10.10 64 bit using "-stdlib=libstdc++ -mmacosx-version-min=10.6". The error is missing std::unique_ptr.

```
tiffimage.cpp:347:8: error: no member named 'unique_ptr' in namespace 'std'
                std::unique_ptr<char[]> formatted;
                ~~~~~^
...
1 warning and 6 errors generated.
```

```
$ g++ -v
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/Applications/
Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.9.sdk/usr/include/c++/4.2.1
Apple LLVM version 5.1 (clang-503.0.40) (based on LLVM 3.4svn)
Target: x86_64-apple-darwin14.3.0
Thread model: posix
```

The fix is to use "-stdlib=libc++ -mmacosx-version-min=10.7", but the minimum system version will be higher - 10.7.

#7 - 25 Apr 2015 20:59 - Robin Mills

Oh, that's odd. I'm also working on MacOS-X. I haven't built on 10.6 (Snow Leopard) for years. I know what's causing that. It's code that I found on StackOverflow. Can I ask you to raise a different bug about building on 10.6 and I'll deal with it tomorrow. I've reproduced your build failure with the command:

```
env MACOSX_DEPLOYMENT_TARGET=10.6 make rebuild ; exiv2 -vV -g date -g time -t svn
```

Perhaps you could check 10.7, 10.8 and 10.9. As they could be building with GCC or Clang. Best to know the extent of the issue. Here's what's building OK for me:

```
773 rmills@rmillsmbp:~/gnu/exiv2/trunk $ date ; sw_vers ; g++ --version ; exiv2 -vV -g date -g time -g svn
```

```
Sat 25 Apr 2015 21:58:44 BST
ProductName:    Mac OS X
ProductVersion:    10.10.3
BuildVersion:    14D136
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/include/c
++/4.2.1
Apple LLVM version 6.1.0 (clang-602.0.49) (based on LLVM 3.6.0svn)
Target: x86_64-apple-darwin14.3.0
Thread model: posix
exiv2 0.24 001800 (64 bit build)
date=Apr 25 2015
time=21:56:27
svn=3732
have_gmtime_r=1
have_timegm=1
774 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

#8 - 26 Apr 2015 09:39 - Robin Mills

- Status changed from Assigned to Resolved

#9 - 08 May 2015 15:55 - Robin Mills

- % Done changed from 0 to 100

#10 - 21 Jun 2015 16:39 - Andreas Huggel

- Status changed from Resolved to Closed