Exiv2 - Patch #1272

Possible issue with temp files being left behind.

19 Jan 2017 21:43 - Ben Touchette

Status:	Closed	**Start date:**	19 Jan 2017
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	basicio	**Estimated time:**	15.00 hours
Target version:	0.26		

Description

While working with a client we noticed that it was possible that some temporary files maybe left behind in the default path in the event of crash. i I don't have a test right now, figured we could be proactive, this patch should address that.

Related issues:

Related to Exiv2 - Feature #747: Direct FILE* access from FileIo interface	**Closed**	**10 Dec 2010**

Associated revisions

Revision 4707 - 15 Feb 2017 20:53 - Robin Mills

#1272 Submitting modified version of Ben's patch.

Revision 4715 - 03 Mar 2017 19:36 - Robin Mills

#1272 Do not use ReaganLargeTiff.tiff in icc-test.out (see #1272 for discussion)

Revision 4716 - 04 Mar 2017 10:50 - Robin Mills

#1272 Supplement to r4715

Revision 4717 - 05 Mar 2017 17:42 - Robin Mills

#1272 Use in-memory temporary files.

History

#1 - 20 Jan 2017 15:45 - Ben Touchette

- File temp_file-1.diff added

This patch supersedes the previous one. Adds a widestring version of temporaryPath.

#2 - 09 Feb 2017 20:37 - Robin Mills

- Category set to jpeg parser

- Status changed from New to Assigned

- Assignee set to Robin Mills

- Target version set to 0.26

#3 - 15 Feb 2017 07:44 - Robin Mills

- % Done changed from 0 to 50

- Estimated time set to 8.00 h

Ben:

Thanks for providing this patch.

The file temp_file-1.diff respects the build flag EXV_UNICODE_PATH which enables wide character paths in MSVC builds. There's something in the patch which is causing the MinGW/32 build to fail to build.

I decided to investigate MinGW + EXV_UNICODE_PATH. This is not good. I do support a nightly build of MinGW/32 for users of Qt. However, I've reached several conclusions about MinGW:

1. MinGW is not fit for any purpose. It's time for Qt to stop using this obsolete (and unfinished) technology. I believe there is new edition "MinGW2" which is posix compliant (and probably based on Cygwin).

2. There really isn't an MinGW/msys/64 environment. You can build 64 bit libraries using a 64 bit version of GCC from MinGW/32.
3. Exiv2 + MinGW + EXV_UNICODE_PATH is unable to build and link samples/exifprint.cpp as it attempts to link ***WinMain@16*** This can be addressed with the command:

```
$ make exifprint CXXFLAGS=-console LDFLAGS=-lmingw32
```

Amazingly, it then fails to link Exiv2:dumpLibraryInfo and Exiv2::ImageFactory::open(...);

```
undefined reference to
`Exiv2::ImageFactory::open(std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >
 const&, bool)'
```

I cannot find the reason for this link failure. The following tool reveals the entry points

```
$ nm -g /usr/local/lib/libexiv2.a | grep ImageFactory | grep open | grep T | cut -d' ' -f 3
```

The unsatisfied external seems to be in the library

```
$ 466 -32- /home/rmills/gnu/exiv2/trunk> c++filt $(nm -g /usr/local/lib/libexiv2.a | grep ImageFactory | g
rep open | grep T | cut -d' ' -f 3)
Exiv2::ImageFactory::open(unsigned char const*, long)
Exiv2::ImageFactory::open(std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >
const&, bool)
Exiv2::ImageFactory::open(std::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, b
ool)
Exiv2::ImageFactory::open(std::auto_ptr<Exiv2::BasicIo>)
467 -32- /home/rmills/gnu/exiv2/trunk>
```

It also grumbles about: ***bad reloc address 0xe in section `.text$_ZNSt9exceptionC2Ev[__ZNSt9exceptionC2Ev]'*** which is in the standard library.

I'm going to reconsider your patch as follows: It has to successfully build the currently supported platforms: MSVC 2005/8/10/12/13/15 (with/without EXV_UNICODE_PATH) + Cygwin + MinGW/32 (without EXV_UNICODE_PATH) + Linux + MacOSX

I know there are issues with your patch as it uses the C run-time library ***remove*** which isn't supported by MSVC. It should be ***_remove/_wremove**** depending on EXV_UNICODE_PATH.

I hope to successfully undertake this work today (2017-02-15)

#4 - 15 Feb 2017 11:27 - Ben Touchette

I had remove working here with MSVC. Interesting mess or something :)

MinGW64 at minimum? it uses a more modern version of gcc. It's what i tested some of these patches with that i've submitted in the last few weeks. I hadn't realized someone would build that antique.

#5 - 15 Feb 2017 11:35 - Robin Mills

- % Done changed from 50 to 70

- Estimated time changed from 8.00 h to 10.00 h

Good News. I got up at 6am and have your patch working on MSVC. I have to go out today and I will test my version of your patch on Cygwin/etc this evening. I hope to have this finished this evening.

Do you have a use case that leaves files in the temporary directory? Several people have mentioned this. I've only observed it when I'm debugging and terminate the program early and therefore don't execute the cleanup code.

#6 - 15 Feb 2017 11:38 - Ben Touchette

Sounds good, no i don't have a use case but my client had some issues with it in their app where it happened mostly in windows to my knowledge.

#7 - 15 Feb 2017 19:46 - Robin Mills

- % Done changed from 70 to 50

- Estimated time changed from 10.00 h to 14.00 h

I've discovered there MSVC test failures on the trunk. I think they've been here for a few weeks. These are "clouding the issue" with your patch. I will fix them of course (probably tomorrow). In the meanwhile, I'm testing my modified version of your patch on several platforms.

The issues are all concerning ICC profiles in the file ReaganLargeTiff.tiff For example:

```
C:\Users\Shared\workspace\Exiv2-trunk@2\label\msvc\test\data\bugfixes-test.out differ
2722c2722,2765
```

```
< ./functions.source: line 6:  9628 Segmentation fault      $VALGRIND $bin$prog "$@"
---
>      334 | 0x8769 ExifTag                    |      LONG |        1 |   1622642 | 1622642
>   STRUCTURE OF TIFF FILE (II): ReaganLargeTiff.tiff
```

The tests icc-test.sh and stdin-test.sh are impacted as they also use that file for ICC Profile Extraction.

#8 - 15 Feb 2017 19:49 - Ben Touchette

Interesting. I hadn't run all the tests, just what they needed when i did that first patch.

#9 - 15 Feb 2017 20:55 - Robin Mills

- % Done changed from 50 to 60

I've submitted a modified variant of your patch. The original code uses DeleteFile and not remove in MSVC. So no particular change required and that explains why it built and passed the test suite for you.

I understand what's wrong in icc_test.sh. On MSVC, writing the ICC profile to stdout is adding 32 mysterious bytes to the profile. This issue #2120 was solved using _binmode() out std::cout. I'll hunt back some builds to discover when/how this has been reintroduced.

```
C:\Users\rmills\gnu\exiv2\trunk\msvc\bin\win32\releasedll>exiv2 -eC- --verbose ReaganLargeTiff.tiff > foo.icc

C:\Users\rmills\gnu\exiv2\trunk\msvc\bin\win32\releasedll>exiv2 -eC  --verbose ReaganLargeTiff.tiff
File 1/1: ReaganLargeTiff.tiff
Writing iccProfile: .\ReaganLargeTiff.icc

C:\Users\rmills\gnu\exiv2\trunk\msvc\bin\win32\releasedll>dir *.icc
 Volume in drive C has no label.
 Volume Serial Number is 0899-EF40

 Directory of C:\Users\rmills\gnu\exiv2\trunk\msvc\bin\win32\releasedll

2017-02-15  20:33         1,613,632 foo.icc
2017-02-15  20:33         1,613,600 ReaganLargeTiff.icc
               2 File(s)      3,227,232 bytes
               0 Dir(s)  25,019,547,648 bytes free

C:\Users\rmills\gnu\exiv2\trunk\msvc\bin\win32\releasedll>
```

Even with the additional bytes in foo.icc, it shouldn't crash of course.

I'm still not sure how/when temporary files are being left on the disk. I'll add some instrumentation code to static std::string tempPath().

std::string tempPath() is not thread safe. On Windows it should use ***InterlockedIncrement*** and declared atomic on other platforms.

So, I've got some work to do to fix this. Priority:
1) Fix the crash and writing the wrong number of bytes to std::cout
2) Make tempPath() thread safe.
3) Instrument tempPath() to find out what's going on.

#10 - 03 Mar 2017 19:24 - Robin Mills

I've done a lot of investigation into ***ReaganLargeTiff.tiff****and****test/icc-test.sh*** and reached the unsatisfactory conclusion that this file is causing the Cygwin/bash test environment distress of some kind which results in a ***segfault*** in the releasedll (but neither release nor debugdll) versions. It doesn't happen from a DOS bat file. It doesn't happen when I source the script from Cygwin/bash. I've reduced the script to the minimum as follows for test/1272.sh:

```
#/bin/bash
cd tmp
cp -f ../data/ReaganLargeTiff.tiff          ReaganLargeTiff.tiff
cp -f ../data/large.icc                     ReaganLargeTiff.icc
../../msvc/bin/win32/releasedll/exiv2 -iC   ReaganLargeTiff.tiff
```

And test/1272.bat

```
pushd  tmp
copy/y ..\data\ReaganLargeTiff.tiff
copy/y ..\data\large.icc                      ReaganLargeTiff.icc
..\..\msvc\bin\win32\releasedll\exiv2.exe -iC ReaganLargeTiff.tiff
popd
```

The error is a "Segmentation Error" which occurs in exiv2(.exe) in src/actions.cpp here:

```
    int Insert::insertIccProfile(const std::string& path,Exiv2::DataBuf& iccProfileBlob) const
```

```
    {
        int rc = 0;
        // test path exists
        if (rc==0 && !Exiv2::fileExists(path, true)) {
            std::cerr << path
                      << ": " << _("Failed to open the file\n");
            rc=-1;
        }

        // read in the metadata
        if ( rc == 0 ) {
            std::cerr << "opening path " << path << std::endl;
            Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(path);
            assert(image.get() != 0);
            image->readMetadata();  // <---- fails to return from here
            // clear existing profile, assign the blob and rewrite image
            image->clearIccProfile();
            if ( iccProfileBlob.size_ ) {
                std::cerr << "about to set the data" << std::endl;
                image->setIccProfile(iccProfileBlob);
            }
            image->writeMetadata();
        }

        return rc;
    } // Insert::insertIccProfile
```

The failure to return is mysterious and smells of stack corruption as I have added trace statements and readMetadata() ran to completion.

It's easy to reproduce this error from the test harness. It's almost impossible to reproduce it with any debugging tools to obtain more information. It's very likely that no user will be inserting a large (1.6mb) ICC profile using an MSVC build from Cygwin. If they report such a calamity, these notes will be useful as a start for further investigation. For the moment, I intend to update the test harness to omit use of ReaganLargeTiff.tiff and this will fix the test harness.

This matter has been given at least 12 hours of investigation and analysis. No further activity is possible into ***ReaganLargeTiff.tiff****and* ***test/icc-test.sh*** at this time.

#11 - 04 Mar 2017 11:55 - Robin Mills

- % Done changed from 60 to 50

- Estimated time changed from 14.00 h to 20.00 h

I'm not convinced that there's an issue to be investigated with temporary files when an application crashes. The application could provide an atExit() handler to clean up. Libraries should not get involved with atExit() handlers which is a process responsibility (and therefore host application responsibility). I'll say more about this later.

The library does contain a thread unsafe function tempPath() which is called by BasicIo::AutoPtr FileIo::temporary() to create a temporary io stream. The concept of tempPath() is horrible and I take the blame, it should be tempFile() and return an open FILE* and pathname. Thread safe. The BasicIo::AutoPtr FileIo::temporary() should call tempFile() and set a flag bool bDeleteOnClose in class FileIo.

A better idea is to never use temporary files and use BasicIo::AutoPtr MemIo::temporary() const instead as it maintains the data in memory which will be cleaned when the BasicIo::AutoPtr goes out of scope.

I am going to investigate removing BasicIo::AutoPtr FileIo::temporary() from the library and using BasicIo::AutoPtr MemIo::temporary() instead.

If it's necessary to keep FileIo::temporary(), I will investigate tempFile() and add the bDelete flag. I could add a BasicIo static function to return a list of all open tempFile() paths which could be called by the application's atExit() handler. I don't intend to add such code for v0.26.

#12 - 04 Mar 2017 12:13 - Ben Touchette

If you do that what happens to classes using remoteio ?

#13 - 04 Mar 2017 13:44 - Robin Mills

- % Done changed from 50 to 80

- Estimated time changed from 20.00 h to 15.00 h

No problem with remoteio. The temporary files are used in the image handlers to create the metadata in a temporary file. This is done within the safety of try/catch. If the metadata is successfully written in temporary storage, it is "transferred" to the "real" file being processed (which may be a remote file). So the temporary file is almost invisible to the remote file. For sure, there is no concept in Exiv2 of a "temporary remote file".

I have removed the API FileIo::temporary(). Every image handler uses BasicIo::MemIo for temporary storage. No temporary FileIo objects are used by the library. Everything is in memory. Incidentally, those files are small as they only contain metadata. Typically 5k-10k. Test suite detects no

issues. Very pleased and happy.

I've moved the horrible/ugly static tempPath() to actions.cpp which is compiled/linked with exiv2(.exe). Whose idea was it to have a horror like that in the library? I've added a mutex (on Unix) and a critical_section (on Windows) to make it thread safe. I strongly dislike anything like this because it can dead-lock ***however*** it's only used by the function metacopy() in actions.cpp in exiv2(.exe) which isn't multi-threaded and is sample code. It's good enough for purpose.

I'll submit the code this evening. It's a nice sunny Saturday afternoon in England and I'm going to do some work in the garden.

#14 - 05 Mar 2017 17:44 - Robin Mills

- Status changed from Assigned to Resolved

- % Done changed from 80 to 100

Fix submitted r4717

I investigated the possibility of tempPath() returning a FILE*. This would provide the lock courtesy of fopen(). However this would require a new FileIo API to pass a FILE*. Too risky at this time in v0.26.

I'll close this in the next few days when it's building and passing the test suite on the buildserver.

#15 - 06 Mar 2017 13:38 - Robin Mills

- Status changed from Resolved to Closed

#16 - 09 Mar 2017 19:56 - Robin Mills

- Category changed from jpeg parser to basicio

Files

temp_files.diff	2.11 KB	19 Jan 2017	Ben Touchette
temp_file-1.diff	4.22 KB	20 Jan 2017	Ben Touchette