Exiv2 - Feature #1176

Validate HUGE file support

31 Mar 2016 14:32 - Robin Mills

Status:	New	**Start date:**	31 Mar 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	testing	**Estimated time:**	0.00 hour
Target version:	1.0		

Description

Although Exiv2 supports a 64 bit build, a lot of the I/O and internal operations are performed using 32 bit integers. As image files (and video in particular) get bigger, it's increasingly important that we support HUGE files (many >2GB and all >4GB).

Related issues:

Related to Exiv2 - Feature #992: Better raw file support and test	**Assigned**	**18 Sep 2014**

History

#1 - 26 Apr 2017 09:47 - Dimitri Schoolwerth

Just a note to say that while I of course appreciate huge file support being looked into I don't think its support should depend on the data model being used, but instead on the target system. A 32-bit system can be perfectly capable of working with large files.

As such it may not be a good idea to have the recent change where size() now returns size_t (r4756) but instead a new typedef could be used, for example Exiv2::FileOffset/file_offset. This would also get rid of conditional constructs like:

```
#if defined(_MSC_VER)
        virtual int seek(int64_t offset, Position pos) = 0;
#else
        virtual int seek(long offset, Position pos) = 0;
#endif
```

For our own purposes we have been using large file support with Exiv2 for a long time, admittedly we do limited things with Exiv2 (just preserving the metadata across image files such as JPEG and TIFF) so for us it was likely much easier to implement. We basically only changed seek(), tell(), and size() to get rid of 2+ GB offset problems.

Also we kept the offset as a signed type. For one because the POSIX off_t (and also, non-POSIX, off64_t) must be signed and I don't like to go against the grain. And also because it had less of an impact on the number of changes compared to using an unsigned type. And while it means that we would still suffer from 2-4 GB trouble because of using a signed 32-bit integer, in practice all our targets (Windows, macOS, and some *nix internally) have been supporting large files for a long time.

#2 - 26 Apr 2017 10:44 - Robin Mills

Thank You very much for your comments. It's good to know that HUGE file support is working adequately for you. For sure, HUGE file support can be made to work on 32-bit processors.

I added the MSVC specific code about 2011 when I converted our MSVC solution/project files to build 64bit. As I was fairly new to the project, I wanted to minimise code changes and this was added because it worked. I wanted to make no changes that could impact the existing Linux code. It's a good example of code that should be carefully inspected and reviewed.

Quickly introducing changes such as r4756 is risky and I would like a more rigorous approach. The changes introduced by r4756 caused "code ripple" which resulted in build issue on MSVC and Linux GCC/5.

```
771 rmills@rmillsmbp:~/gnu/exiv2/trunk $ svn log . | grep -e '#1175'
#1175 more cast ripples on GCC5/Linux
#1175 Correction to r4597 Additional file (which, in error, I didn't to commit)
#1175 Correction to r4756 Another three casts required to build with Visual Studio (size_t code ripple)
#1175 I'm going to accept the recommendation to change BasicIo::size() to return size_t.
This passes the test suite.
The only "ripple" outside of basicio is to iotest.cpp.
This change enables several casts to be removed.
#1175.  Thanks to LaserSoft for reporting this and providing a patch.
#1175.  Thank You, LaserSoft, for reporting this and providing the patch.
772 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

Your suggestion to have an offset_t is helpful. I would like to study HUGE file IO using standard libraries. It would be good to make basicio a library that could be used in other projects (and statically linked into Exiv2).

I'm not going to do anything about this at the moment as my aim is to release v0.26 in the next few days.

#3 - 26 Apr 2017 13:37 - Dimitri Schoolwerth

Thank you for the explanation. Completely understood about not changing anything now with a release pending of course. Just wanted to add my two cents before any bigger changes would be made. And I forgot to mention I wouldn't mind looking into this as well (just not right now), probably by starting with running the existing test suites and then adding one that fails on large files.

#4 - 26 Apr 2017 13:45 - Robin Mills

I would be delighted to accept your help. I suggest you start on basicio.cpp (and http.cpp) and validate them on large files. That's quite an easy task. Once you get involved with reading images, you have code for each image type to consider.

We can chat when you're ready to spend time on this.