# Exiv2 - Support #1290
# write exif to a libgphoto2 buffer image

21 Apr 2017 10:09 - Nacho Sánchez Moreno

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 21 Apr 2017 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Robin Mills | **% Done:** | 100% |
| **Category:** | not-a-bug | **Estimated time:** | 7.00 hours |
| **Target version:** | 0.26 | | |

**Description**

Hello,

I'm trying to write exif data to a libgphoto2 buffer.

```
unsigned long int file_size = 0;
const char *file_data = NULL;

with gp_file_get_data_and_size(file, &file_data, &file_size) i retrive the buffer and size for an NEF image (nikkon d810)
After i try:

Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open((const Exiv2::byte*)file_data, file_size);

Exiv2::ExifData exifData;
exifData["Exif.Image.Software"] = this->_metadataExif.Software;
exifData["Exif.Image.Artist"] = this->_metadataExif.Artist;
exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright;

// Set EXIF data and write it to the file
image->setExifData(exifData);
image->writeMetadata();

file_size = image->io().size();
image->io().seek(0,Exiv2::BasicIo::beg);
Exiv2::DataBuf buff = image->io().read(file_size);

// -------------------------------------------------------------
// Write the buff to disk
FILE * filen = ::fopen(fname.c_str(), "w");
::fwrite(buff.pData_, file_size, 1, filen);
::fclose(filen);

gp_file_free(file);
```

them, the exif metadata information is ok, but the file is corrupted.

where is the problem?
thanks

**History**

**#1 - 21 Apr 2017 11:35 - Robin Mills**

*- Category set to not-a-bug*

*- Status changed from New to Assigned*
*- Assignee set to Robin Mills*
*- Target version set to 0.26*

Nacho

I don't know libgphoto. I believe you are using Exiv2 to manipulate the image in memory and then use fopen/fwrite/fclose to write the image to disk. I believe this should work, so without your libgphoto code, I can't say why this isn't working for you.

I've written the following code to simulate libgphoto using our old friends malloc/stat/fread:

```
// g++ testExiv2MemIO.cpp -lexiv2 -o testExiv2MemIO

#include <exiv2/exiv2.hpp>

#include <sys/stat.h>
#include <unistd.h>

int main(int argc,const char* argv[])
{
   const char*          program  = argv[0];
   if ( argc != 3 )     return printf("syntax %s in-path out-path\n",program);

   const char*          path_in  = argv[1];
   const char*          path_out = argv[2];

   struct stat          buf      ;
   if (    stat(path_in,&buf) ) return printf("path %s does not exist\n",path_in);

   // allocate a memory buffer into which to read path_in
   unsigned long int    file_size = buf.st_size;
   const char*          file_data = (const char*) ::malloc(file_size);
   if ( !file_data)     return printf("unable to allocate %lu bytes for %s",file_size,path_in);

   // read path_in into a memory buffer
   FILE*                file_in   = ::fopen(path_in,"rb");
   if ( !file_in )      return printf("unable to open %s\n",path_in);
   int                  n         = fread( (void*) file_data,1,file_size,file_in);
   printf("fread %ld bytes from %s\n",file_size,path_in);

   // create an Exiv2 image from the buffer
   Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open((const Exiv2::byte*)file_data, file_size);

   // modify the Exiv2 metadata
   Exiv2::ExifData      exifData;
   exifData["Exif.Image.Software"] = program;
   exifData["Exif.Image.Artist"]   = program;

   // write metadata to memory
   image->setExifData(exifData);
   image->writeMetadata();

   // recover Exiv2 image buff
```

```
    file_size = image->io().size();
    image->io().seek(0,Exiv2::BasicIo::beg);
    Exiv2::DataBuf buff = image->io().read(file_size);

    // Write buff to disk
    FILE *              file_out = ::fopen(path_out, "w");
    n = ::fwrite((void*)  buff.pData_, 1,file_size, file_out);
    ::fclose(file_out);
    printf("fwrite %ld bytes to %s\n",file_size,path_out);

    ::free((void*)file_data);

    return 0;
}
```

This seems to work fine:

```
559 rmills@rmillsmbp:~/temp $ cp ~/Pictures/Wallpapers/Stonehenge\,England.jpg Stonehenge.jpg
560 rmills@rmillsmbp:~/temp $ exiv2 -pa --grep Software --grep Artist Stonehenge.jpg
Exif.Image.Software                  Ascii      10  Ver.1.00
561 rmills@rmillsmbp:~/temp $ ./testExiv2MemIO Stonehenge.jpg S.jpg
fread 6780056 bytes from Stonehenge.jpg
fwrite 6764223 bytes to S.jpg
562 rmills@rmillsmbp:~/temp $ exiv2 -pa --grep Software --grep Artist S.jpg
Exif.Image.Software                  Ascii      17  ./testExiv2MemIO
Exif.Image.Artist                    Ascii      17  ./testExiv2MemIO
563 rmills@rmillsmbp:~/temp $
```

I've tried the code on a .NEF from my Nikon D5300:

```
564 rmills@rmillsmbp:~/temp $ cp /Photos/2016/Raw/DSC_0001.NEF .
565 rmills@rmillsmbp:~/temp $ ./testExiv2MemIO DSC_0001.NEF D.NEF
fread 22330428 bytes from DSC_0001.NEF
fwrite 18158964 bytes to D.NEF
566 rmills@rmillsmbp:~/temp $ exiv2 -pa --grep Software --grep Artist D.NEF
Exif.Image.Software                  Ascii      17  ./testExiv2MemIO
Exif.Image.Artist                    Ascii      17  ./testExiv2MemIO
567 rmills@rmillsmbp:~/temp $
```

I suspect the issue lies in the libgphoto buffer. A modified strategy for you is to write the libgphoto buffer to disk. Then use Exiv2 to open, read, modify and write your file. Can you try this?

If you're still stuck, please share your libgphoto code and I will download libgphoto and build your code. When we are *"both on the same page"*, I believe we'll quickly resolve this matter.

Robin

#### #2 - 21 Apr 2017 12:10 - Nacho Sánchez Moreno

Hi Robin,

Thanks a lot for your answer.

Yes, i tryed too with the strategy of to write the libgphoto2 to disk, and i've had some problems depending the code, with errors like:

basicio.cpp:598: virtual long int Exiv2::FileIo::write(Exiv2::BasicIo&): Assertion `p_->fp_ != 0' failed when i use fileio.write(image->io());

or:

Program received signal SIGSEGV, Segmentation fault.
0x766e5404 in fwrite () from /lib/arm-linux-gnueabihf/libc.so.6 when i use ::fwrite(buff.pData_, buffersize, 1, fileio);

Anyway I will continue to insist on the initial solution, ie using the image in memory because i've retrive the image from the camera previously and i think that is the moment to write de exif data before to write to disk.

thanks again

#### #3 - 21 Apr 2017 12:23 - Robin Mills

*- Status changed from Assigned to Resolved*
*- % Done changed from 0 to 100*
*- Estimated time set to 1.00*

You are asking me to debug a scenario that I cannot reproduce.  Can I ask you to strip your libgphoto code to the minimum and let me build/reproduce your error.

I'm going out for three hours.  Drop me your code ***and together***, we can fix this today.

#### #4 - 21 Apr 2017 12:41 - Nacho Sánchez Moreno

Hi,

Thank you very much,

I also have to leave, and probably until Tuesday can not continue. I think I'm not taking precaution with memory somewhere. Because the files it generates them, but all of the same size and corrupted, but with the exif data.

At the finaly of the code i've comment some tries about the second scenario, ie save to disk previously

Basically the code is:

```
CameraFile *file;

ret = gp_file_new(&file);

if (ret != GP_OK)
   continue;

// get de photo from the camera
ret = gp_camera_file_get(this->_camera, cPath.folder, cPath.name, GP_FILE_TYPE_NORMAL, file, this->_ctx);

if (ret != GP_OK)
   continue;

// get buffer and size
ret = gp_file_get_data_and_size (file, &file_data, &file_size);
if (ret != GP_OK)
   continue;
```

```
try {
    // Open image file (WITH YOUR NOTES)
    Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open((const Exiv2::byte*)file_data, file_size);

    Exiv2::ExifData exifData;
    exifData["Exif.Image.Software"] = this->_metadataExif.Software; // IT'S OK
    exifData["Exif.Image.Artist"] = this->_metadataExif.Artist; // IT'S OK
    exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright; // IT'S OK

    // write metadata to memory
    image->setExifData(exifData);
    image->writeMetadata();

    // recover Exiv2 image buff
    file_size = image->io().size();
    image->io().seek(0,Exiv2::BasicIo::beg);
    Exiv2::DataBuf buff = image->io().read(file_size);

    // The file's name:
    fname = this->_pName + "/" + fname;

    // -------------------------------------------------------------
    // Write the buff to disk
    FILE * file_out = ::fopen(fname.c_str(), "w");
    ::fwrite((void*) buff.pData_, 1, file_size, file_out);
    ::fclose(file_out);
}
catch (Exiv2::AnyError& e) {
    std::cout << "Caught Exiv2 exception '" << e << "'\n";
}

fname = this->_pName + "/" + fname;

//ret = gp_file_save(file, fname.c_str());

gp_file_free(file);

//if (ret != GP_OK)
//    continue;

/*
Exiv2::DataBuf buf = Exiv2::readFile(fname.c_str());
Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(buf.pData_, buf.size_);
image->readMetadata();
Exiv2::ExifData &exifData = image->exifData();
exifData["Exif.Image.Software"] = this->_metadataExif.Software;
exifData["Exif.Image.Artist"] = this->_metadataExif.Artist;
exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright;
image->writeMetadata();

string fname1 = this->_pName + "/" + "_try_" + fname;

Exiv2::FileIo fileio(fname1.c_str());
```

```
fileio.open();
fileio.write(image->io());
*/

/*
file_size = image->io().size();
image->io().seek(0,Exiv2::BasicIo::beg);
Exiv2::DataBuf buff = image->io().read(file_size);

FILE * fileio = ::fopen(fname1.c_str(), "w");
::fwrite(buff.pData_, file_size, 1, fileio);
::fclose(fileio);
*/

file_size = 0;
file_data = NULL;
```

#### #5 - 21 Apr 2017 12:44 - Nacho Sánchez Moreno

Sorry,

Strange things have happened with the format

#### #6 - 21 Apr 2017 12:52 - Robin Mills

Thanks for the code. I'll look at this tonight. We'll successfully fix this.

Don't worry about the format. You should enclose code: **<pre>** ... **</pre>**. I'll edit your comments and fix the formatting later.

Have a nice day and weekend. Speak later.

#### #7 - 21 Apr 2017 14:10 - Robin Mills

*- % Done changed from 100 to 80*
*- Estimated time changed from 1.00 to 4.00*

I think I know what's wrong with this. I haven't debugged your code, I've deduced it by inspection.

Only a tiny part of an image contains metadata. Most of the data in an image is pixels. When you use ***image->readMetadata()***, Exiv2 finds the metadata and puts it into the Exiv2::ExifData vector. It is possible for ***image->readMetadata()*** to be successful on a corrupted image. However ***image->writeMetadata()*** could fail if the pixel data in the image is corrupted. It's an uncommon scenario for images to be created from memory buffers. Images are normally created by reading good images into memory.

I think there is something wrong with the data ***file_data*** and ***file_size***. Can I ask you to write that data immediately to path ***foo.image*** (using a variant of your code: ***fileio ::fopen ::write ::fclose***). If ***foo.image*** is corrupt, it has **not** been corrupted by Exiv2 as no Exiv2 code has seen that data.

You can analyse ***foo.image*** with the command: $ exiv2 -pR foo.image
Please attach ***foo.image*** to this report.

My earlier suggestion to write to disk and use ***Exiv2::ImageFactory::open(path)*** has been ineffective because the image written to disk is corrupted. Opening an image is essentially (inside Exiv2) ***::fopen*** followed by ***readMetadata()***. The image is still corrupt and will again be detected by your call to ***image->writeMetadata()***

**#8 - 24 Apr 2017 08:13 - Robin Mills**

*- Status changed from Resolved to Closed*

*- % Done changed from 80 to 100*

**#9 - 25 Apr 2017 07:32 - Nacho Sánchez Moreno**

Hello,

I tried again with the program.

with:

```
// fname_out is the name file from the camera to disk
string fname_out = this->_pName + "/" + fname;

// save the file from the buffer to disk (IT'S OK)
ret = gp_file_save(file, fname_out.c_str());

gp_file_free(file);

Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(fname_out.c_str());

image->readMetadata();
Exiv2::ExifData &exifData = image->exifData();

exifData["Exif.Image.Software"] = this->_metadataExif.Software;
exifData["Exif.Image.Artist"] = this->_metadataExif.Artist;
exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright;
// write metadata to memory
image->setExifData(exifData);
image->writeMetadata();

// fname1 is a name for the file with exif metadata
string fname1 = this->_pName + "/" + "_2_" + fname;

// recover Exiv2 image buff (I'VE TESTED THAT THE SIZE IS OK)
file_size = image->io().size();
// the program fails here:
image->io().seek(0, Exiv2::BasicIo::beg);
// the error is:  basicio.cpp:850: virtual int Exiv2::FileIo::seek(long int, Exiv2::BasicIo::Position): Assertion `p_->fp_ != 0' failed.

Exiv2::DataBuf buff = image->io().read(file_size);

FILE * file_out = ::fopen(fname1.c_str(), "w");
int n = ::fwrite((void*)  buff.pData_, 1,file_size, file_out);
::fclose(file_out);

printf("fwrite %ld bytes to %s\n", file_size, fname1.c_str());
```

i've tried with another version:

```
// fname_out is the name file from the camera to disk
string fname_out = this->_pName + "/" + fname;

// save the file from the buffer to disk (IT'S OK)
ret = gp_file_save(file, fname_out.c_str());

gp_file_free(file);

Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(fname_out.c_str());

image->readMetadata();
Exiv2::ExifData &exifData = image->exifData();

exifData["Exif.Image.Software"] = this->_metadataExif.Software;
exifData["Exif.Image.Artist"] = this->_metadataExif.Artist;
exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright;
// write metadata to memory
image->setExifData(exifData);
image->writeMetadata();

// fname1 is a name for the file with exif metadata
string fname1 = this->_pName + "/" + "_2_" + fname;

Exiv2::FileIo file_out(fname1.c_str());
file_out.open();
// the program fails here:
file_out.write(image->io());
// the error is the same:  basicio.cpp:598: virtual long int Exiv2::FileIo::write(Exiv2::BasicIo&): Assertion `p_->fp_ != 0' failed.
```

**#10 - 25 Apr 2017 07:54 - Robin Mills**

Nancho

Please attach your saved file to this issue.  I believe you are crashing because it does not contain a valid image.

Robin

**#11 - 25 Apr 2017 08:13 - Nacho Sánchez Moreno**

*- File src.zip.001 added*

*- File src.zip.002 added*

*- File src.zip.003 added*

Ok,

I upload two images:

DSC_2172_gp.NEF is the original image created with nikon d810 and saved in disk using gp_file_save(file, fname_out.c_str());
and _2_DSC_2172_gp is the image corrupted but with the exif data.

I not sure that the process could have problems after gp_file_save and gp_file_free for the image is not in the disk totally before open again with Exiv2::ImageFactory::open(fname_out.c_str());

It's because i think that the best proccess is manage the buffer in memory that to save to disk instead

i used peazip to compress

(sorry for my english)

**#12 - 25 Apr 2017 10:11 - Robin Mills**

*- Category changed from not-a-bug to image format*
*- Status changed from Closed to Assigned*
*- % Done changed from 100 to 50*
*- Estimated time changed from 4.00 to 10.00*

Thanks for the files. Good News. I can now reproduce your result with testExiv2MemIO.cpp (my code above). And with my Raw file from my D5300.

You've send me two files. DSC_2172_gp.NEF is good. _2_DSC_2172_gp.NEF is corrupted. 579 rmills@rmillsmbp:~/Downloads $ cat src.zip.001 src.zip.002 src.zip.003 > src.zip
580 rmills@rmillsmbp:~/Downloads $ unzip src.zip
Archive: src.zip
inflating: DSC_2172_gp.NEF
inflating: _2_DSC_2172_gp.NEF
581 rmills@rmillsmbp:~/Downloads $ dir src
-rwxr-xr-x+ 1 rmills staff 32M Apr 25 08:41 ./src/DSC_2172_gp.NEF*
-rwxr-xr-x+ 1 rmills staff 30M Apr 25 08:41 ./src/_2_DSC_2172_gp.NEF*
582 rmills@rmillsmbp:~/temp $
Running testExiv2MemIO on your good file creates a file that is corrupted.
601 rmills@rmillsmbp:~/Downloads $ cd ~/temp
602 rmills@rmillsmbp:~/temp $ ./testExiv2MemIO ~/Downloads/src/DSC_2172_gp.NEF ungood.NEF
fread 33007232 bytes from /Users/rmills/Downloads/src/DSC_2172_gp.NEF
fwrite 31002552 bytes to ungood.NEF
603 rmills@rmillsmbp:~/temp $ ls -alt ungood.NEF
-rw-r--r-- 1 rmills staff 31002552 Apr 25 10:35 ungood.NEF
604 rmills@rmillsmbp:~/temp $ $ ls -alt ungood.NEF
-rw-r--r-- 1 rmills staff 31002552 Apr 25 10:35 ungood.NEF
604 rmills@rmillsmbp:~/temp $ exiv2 -pa ungood.NEF
Exif.Image.Software Ascii 17 ./testExiv2MemIO
Exif.Image.Artist Ascii 17 ./testExiv2MemIO
Exif.Image.SubIFDs Long 1 86
Exif.SubImage1.NewSubfileType Long 1 Primary image
Exif.SubImage1.ImageWidth Long 1 4820
Exif.SubImage1.ImageLength Long 1 3216
Exif.SubImage1.BitsPerSample Short 1 14
Exif.SubImage1.Compression Short 1 Uncompressed
Exif.SubImage1.PhotometricInterpretation Short 1 Color Filter Array
Exif.SubImage1.StripOffsets Long 1 312
Exif.SubImage1.SamplesPerPixel Short 1 1

```
Exif.SubImage1.RowsPerStrip              Long       1  3216
Exif.SubImage1.StripByteCounts           Long       1  31002240
Exif.SubImage1.XResolution               Rational   1  300
Exif.SubImage1.YResolution               Rational   1  300
Exif.SubImage1.PlanarConfiguration       Short      1  1
Exif.SubImage1.ResolutionUnit            Short      1  inch
Exif.SubImage1.CFARepeatPatternDim       Short      2  2 2
Exif.SubImage1.CFAPattern                Byte       4  0 1 1 2
Exif.SubImage1.SensingMethod             Short      1  2
605 rmills@rmillsmbp:~/temp $
```

Good progress. I now have something to debug. I've revisited my test on a file from my Nikon D5300. I only inspected the output for Artist and Copyright. The NikonD5300 file has been corrupted. The issue appears to be when we rewrite the NEF file.

However, I can rewrite the the file with the exiv2 application: 609 rmills@rmillsmbp:~/temp $ cp /Photos/2016/RAW/DSC_0001.NEF .

```
610 rmills@rmillsmbp:~/temp $ many exiv2
611 rmills@rmillsmbp:~/temp $ ls -alt DSC_0001.NEF ; exiv2 -M'set Exif.Image.Artist Robin' DSC_0001.NEF ; ls -alt DSC_0001.NEF
-rwxr-xr-x 1 rmills staff 22330428 Apr 25 10:54 DSC_0001.NEF
-rwxr-xr-x 1 rmills staff 22328818 Apr 25 10:56 DSC_0001.NEF
612 rmills@rmillsmbp:~/temp $ open DSC_0001.NEF
613 rmills@rmillsmbp:~/temp $
```

And using your file: 618 rmills@rmillsmbp:~/temp $ ls -alt DSC_2172_gp.NEF;exiv2 -M'set Exif.Image.Artist Robin' DSC_2172_gp.NEF;ls -alt DSC_2172_gp.NEF

```
-rwxr-xr-x 1 rmills staff 33007232 Apr 25 10:59 DSC_2172_gp.NEF
-rwxr-xr-x 1 rmills staff 32996840 Apr 25 11:00 DSC_2172_gp.NEF
619 rmills@rmillsmbp:~/temp $ open DSC_2172_gp.NEF
620 rmills@rmillsmbp:~/temp $
```

Conclusion: These files can be successfully updated, so we're using the API incorrectly in some way.

I would like to defer this issue for the moment. My priority for this week is to release v0.26. I deferred the release last week to work on a presentation about Exiv2 for a conference in Rio. And worked on a couple of other issues #1291 and #1175. If v0.26 is going to ship this week, I will have to put your issue to the side for the moment. Perhaps you could say something about the urgency of this matter to you.

https://www.youtube.com/watch?v=3Fv57Lbhmqg

#### #13 - 25 Apr 2017 12:35 - Robin Mills

*- Category changed from image format to not-a-bug*

*- Status changed from Assigned to Closed*

*- % Done changed from 50 to 100*

*- Estimated time changed from 10.00 to 6.00*

More good news. I have testExiv2MemIO.cpp working by adding a couple of lines of code. These are lines of code that you already have.

```
Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open((const Exiv2::byte*)file_data, file_size);
    image->readMetadata(); // <--- add this

    // modify the Exiv2 metadata
    Exiv2::ExifData&      exifData  = image->exifData();  // <--- add this
    exifData["Exif.Image.Software"] = program;
    exifData["Exif.Image.Artist"]   = program;
```

I can't solve this issue until we can both reproduce the fault with identical code and identical test files. Can I ask you to confirm that

testExiv2MemIO.cpp is working for you with those changes? 735 rmills@rmillsmbp:~/temp $ g++ testExiv2MemIO.cpp -lexiv2 -o testExiv2MemIO
736 rmills@rmillsmbp:~/temp $ ./testExiv2MemIO ~/Downloads/src/DSC_2172_gp.NEF foo.NEF
fread 33007232 bytes from /Users/rmills/Downloads/src/DSC_2172_gp.NEF
fwrite 32996846 bytes to foo.NEF
737 rmills@rmillsmbp:~/temp $ ls -alt ~/Downloads/src/DSC_2172_gp.NEF foo.NEF
-rw-r--r-- 1 rmills staff 32996846 Apr 25 13:22 foo.NEF
-rwxr-xr-x+ 1 rmills staff 33007232 Apr 25 08:41 /Users/rmills/Downloads/src/DSC_2172_gp.NEF
738 rmills@rmillsmbp:~/temp $

#### #14 - 25 Apr 2017 13:37 - Robin Mills

*- Estimated time changed from 6.00 to 7.00*

I've had lunch and looked again at your code. The first branch of your code is working too hard. If you open a path with ImageFactory, the image will be written to disk when the image is closed or goes out of scope. You don't need to seek, copy the buffer or anything like that. It's as simple as this:

```
Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(path);
image->readMetadata();

// modify the Exiv2 metadata
Exiv2::ExifData&      exifData  = image->exifData();
exifData["Exif.Image.Software"] = program;
exifData["Exif.Image.Artist"]   = program;
image->writeMetata();
```

The second branch of your code is failing because you haven't opened the image for write! Exiv2::FileIo file_out(fname1.c_str());

```
file_out.open(); // "wb" to write
// the program fails here:
file_out.write(image->io());
```

I wrote testExiv2MemIO.cpp to illustrate manipulating in-memory images and how to write them to disk. I thought that would match your libgphoto2 workflow.

Here's a simple example that only involves modifying an image that is already on disk: // g++ exiv2SetArtist.cpp -lexiv2 -o exiv2SetArtist

```
#include <exiv2/exiv2.hpp>

int main(int argc,const char* argv[])
{
   const char*          program  = argv[0];
   if ( argc != 2 )     return printf("syntax %s path\n",program);

   const char*          path     = argv[1];

   // open the image
   Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(path);
   image->readMetadata();

   // modify the Exiv2 metadata
   Exiv2::ExifData&      exifData  = image->exifData();
   exifData["Exif.Image.Software"] = program;
```

```
    exifData["Exif.Image.Artist"]   = program;
    image->writeMetadata();

    return 0;
}
```

Build and test it:

```
741 rmills@rmillsmbp:~/temp $ g++ exiv2SetArtist.cpp -lexiv2 -o exiv2SetArtist
742 rmills@rmillsmbp:~/temp $ cp ~/Downloads/src/DSC_2172_gp.NEF .
743 rmills@rmillsmbp:~/temp $ exiv2 -pa DSC_2172_gp.NEF | wc
   210     997   14561
744 rmills@rmillsmbp:~/temp $ ./exiv2SetArtist DSC_2172_gp.NEF
745 rmills@rmillsmbp:~/temp $ exiv2 -pa DSC_2172_gp.NEF | wc
   210     998   14540
746 rmills@rmillsmbp:~/temp $ exiv2 -pa -g Artist -g Software DSC_2172_gp.NEF
Exif.Image.Software                  Ascii      17  ./exiv2SetArtist
Exif.Image.Artist                    Ascii      17  ./exiv2SetArtist
747 rmills@rmillsmbp:~/temp $
```

#### #15 - 25 Apr 2017 18:48 - Robin Mills

Returning to the code you presented at the start of this issue. I think we now have the issue identified and you should make three changes to your code:

```
unsigned long int file_size = 0;
const char *file_data = NULL;

with gp_file_get_data_and_size(file, &file_data, &file_size) i retrive the buffer and size for an NEF image (nikkon d810)
After i try:

Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open((const Exiv2::byte*)file_data, file_size);

Exiv2::ExifData exifData;
exifData["Exif.Image.Software"] = this->_metadataExif.Software;
exifData["Exif.Image.Artist"] = this->_metadataExif.Artist;
exifData["Exif.Image.Copyright"] = this->_metadataExif.Copyright;

// Set EXIF data and write it to the file
image->setExifData(exifData);
image->writeMetadata();

file_size = image->io().size();
image->io().seek(0,Exiv2::BasicIo::beg);
Exiv2::DataBuf buff = image->io().read(file_size);

// -------------------------------------------------------------
// Write the buff to disk
FILE * filen = ::fopen(fname.c_str(), "w");
::fwrite(buff.pData_, file_size, 1, filen);
::fclose(filen);

gp_file_free(file);
```

The changes are:

1) You need image->readMetadata() before attempting to manipulate the metadata in the image.

2) Get a reference to the metadata in the image with: Exiv2::ExifData& exifData = image->exifData();

3) Because you are operating on a reference to the metadata, I don't think you need the statement: image->setExifData(...)

#### #16 - 26 Apr 2017 06:23 - Nacho Sánchez Moreno

thanks a lot

### Files

| | | | |
|---|---|---|---|
| src.zip.001 | 20 MB | 25 Apr 2017 | Nacho Sánchez Moreno |
| src.zip.002 | 20 MB | 25 Apr 2017 | Nacho Sánchez Moreno |
| src.zip.003 | 11.9 MB | 25 Apr 2017 | Nacho Sánchez Moreno |