Exiv2 - Feature #505

Ability to extract some very basic pieces of information from images

19 Feb 2007 07:25 - Andreas Huggel

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

Description

Hi,

I'd like to request a feature for a future version of exiv2:
the ability to extract some very basic pieces of information
about an image even from files that don't carry any metadata
(Exif, IPTC or XMP). I'm thinking about basic attributes like:

- pixel width
- pixel height
- image depth (bits per pixel)
- samples per pixel
- bits per sample

All of the above attributes can be extracted from the Exif metadata,
but not all image files have Exif metadata.

It could be argued that this goes beyond the scope of a **metadata**
library like libexiv2, and that extraction of the above information
could be easily (?) accomplished using a separate parser.

But for a file format like JPEG, this means doing some work twice,
namely, scanning the file for JPEG markers. A single-pass scan could
both locate APP1, APP13 and COM markers (for Exif/XMP/IPTC metadata
and comments) and the first SOFn marker (for width/height/depth).

Scanning the same file twice (once for libexiv2 and once for
extracting width/height/depth using a separate parser) may seem
like no big deal, but there are scenarios in which accessing the
image file can be expensive -- think of a file living on a remote
file system or on a HTTP server.

-- marco

Related issues:

Related to Exiv2 - Feature #576: 24x36 equivalent focal length	**Closed**	
Related to Exiv2 - Feature #618: Easy access to information which may be in d...	**Closed**	**07 Mar 2009**

History

#1 - 19 Feb 2007 07:35 - Andreas Huggel

Using an interface similar to the existing metadata containers but with all information in one container, these additional items could be added with keys like

Image.Info.Width
Image.Info.Height
...

Further, I see these as read-only items. Or does it make any sense at all to allow users to modify them?

#2 - 19 Feb 2007 08:11 - Marco Piovanelli

Using the Image.Info.* namespace for "intrinsic" metadata like image width, height and depth sounds to me like exactly the right thing to do. Also, I agree with you that intrinsic metadata should be read-only.

#3 - 17 Dec 2008 09:32 - Andreas Huggel

- Target version set to 1.0

#4 - 19 May 2009 20:19 - Robin Mills

The request by Marco mentions "... there are scenarios in which accessing the image file can be expensive -- think of a file living on a remote file system or on an HTTP server."

I believe it would be quite easy to derive a new HttpIo class from BasicIo which can perform (readonly) random access to a file on a web server. Most web servers these days support HTTP 1.1 "Content-Range". So it's not necessary to read the complete file (except in a last resort).

A command such as exiv2 -pt http://clanmills.com/robin.jpg would provide exif data from the net just as exiv2 -pt /Users/rmills/clanmills/robin.jpg does from the disk/lan. The difference would be when the image factory creates the IO object. As the image factory lives in libexiv2 - all exiv2 applications would acquire internet IO.

An elegant refinement would be to provide an IO factory which can create a BasicIo instance appropriate for the "path". So "/home/this/that/other.jpg" creates a FileIo instance and "http://bla/bla" creates an HttpIo instance. Who knows if we'd ever require ftp, smb or other protocols.

The addition of an HttpIo class is a different project from Marco's request. However I'm willing to implement the HttpIo class using public domain HTTP client/socket code. A built switch can be provided to build or omit this code from the library.

#5 - 19 May 2009 23:55 - Andreas Huggel

Yes, it's possible. In fact, Marco had implemented such a class some time ago for his own use and shared it with me but I didn't check it in as I didn't think it is of enough interest.

Just to spin the thought further, an alternative to implementing a low level interface to each different protocol would be to add support for existing abstracted file I/O frameworks like GIO and KIO and and let these deal with the details of the low-level protocols. That also would make it easier for Gnome and KDE applications to use Exiv2. That's something I'd find exciting, although it would still have to remain outside of the main Exiv2 library (or else Gnome apps would suddenly depend on KDE stuff because of Exiv2 and vice versa - we wouldn't get popular that way :).

There is a snag though: Memory mapping. Exiv2 uses memory mapping to read metadata from almost all file formats other than JPEG and for "non-intrusive" writing it reads the complete file into memory and updates it there directly. In the future (in the unstable branch right now, since r1754 and r1755 for #617) it will use memory mapped files also for "non-intrusive" writing to TIFF-like files, so that the file doesn't need to be loaded completely anymore). If there is a good way to deal with that, I'll be all for it.

#6 - 20 May 2009 00:08 - Gilles Caulier

Some tips from a KDE developer :

KIO is thread safe. It's a separated process in fact. We use it in digiKam to query a sqlite database which is not thread safe.

But there is some problem with KIO : you don't have any control about execution priority as with a thread. Also, debugging a KIO-slave is very complicated. This is why i prefer to use multi-threading, and in digiKam this is the way that we use to play with Exiv2.

Gilles Caulier

#7 - 20 May 2009 19:15 - Robin Mills

I didn't know about GIO and KIO. KIO's looks very interesting. I was thinking "curl" as I believe it provides a simple API for http/https/ftp/ftps - however I don't yet hold any preference for the implementation technology.

I'm not overly concerned about the build/dependancy implications. We can use a build switch (as we do for boost/organize). However that's a matter for Andreas - exiv2 is his library.

Memory mapping's an interesting puzzle. The thing that's nice about the read IO model (open/read/seek/tell/close) is that you can use on-demand page loading from the server. Now if there was a way to get an exception when you access memory, we could demand page the data from the server. The ugly C solution is to put a macro on every memory access - so instead of saying char* x = pMappedFile+count, we could do something like char* x = SAFE (pMappedFile+count,bytes) where bytes is an indicator of the number of bytes when you wish to read (doesn't need to be accurate, simply safe).

Perhaps a MemoryMappedFile class in which we overload the + operator will work. The class can have a member size_t bytes_ which defines the safe/guaranteed data zone following any access. For reading TIFF/JPEG dictionaries bytes_ = 8k might be fine. So when you calculate a memory address, you guarantee that address has good data (and the following 8k/bytes_). Cool, eh?

So I think "Yes" we can have read-only Memory Mapped Files which are demand paged from a server. The 'loaded page mask' could be used to refresh the server for a writable file. However in the first instance, I think this can be "readonly" feature.

Negative input welcome. In fact very welcome - you'll save me wasting a beautiful June weekend on this while Ali is in England visiting the grand-kids.

#8 - 20 May 2009 21:28 - Andreas Huggel

> I'm not overly concerned about the build/dependancy implications. We can use a build switch (as we do for boost/organize). However that's a matter for Andreas - exiv2 is his library.

If there ever is something like Exiv2:KioIo or Exiv2::GioIo derived from Exiv2::BasicIo (silly names maybe but you get what I mean) these extensions would definitely have to live in their own little libraries, which would depend on libexiv2 and the relevant KDE or Gnome libraries respectively (or else we end up with libexiv2-kde and libexiv2-gnome).

> so instead of saying char* x = pMappedFile+count, we could do something like char* x = SAFE (pMappedFile+count,bytes) where bytes is an indicator of the number of bytes when you wish to read (doesn't need to be accurate, simply safe).

If we are looking at such an interface we might as well replace the BasicIo::mmap/munmap calls with something like this and then only use mmap/munmap internally in FileIo and something else to implement this new interface in GioIo etc. Do you think that's feasible?

> So I think "Yes" we can have read-only Memory Mapped Files which are demand paged from a server. The 'loaded page mask' could be used to refresh the server for a writable file. However in the first instance, I think this can be "readonly" feature.

That's fine assuming that write support can be added later, without having to reinvent the wheel again.

> Negative input welcome. In fact very welcome - you'll save me wasting a beautiful June weekend on this while Ali is in England visiting the grand-kids.

The buts... Well this sounds like a cool thing. It's not so small though (a lot of existing code will need to be reviewed and modified for the new interface), I doubt one weekend will be enough - but I think you said your wife will be away a bit longer didn't you :)

My main question is whether there is a need for this in the real world or whether the time can be better spent for things that the people out there are really looking for. (We're discussing this completely off-topic of the original feature request...) I think we should have a use-case / test-case in the form of a project that would benefit from such a feature.

Gilles already commented for digiKam, I suggest you seek input from other projects too, e.g., gthumb (try Michael Chudobiak: mjc at avtechpulse dot com) - If it uses GIO, would gthumb benefit from Exiv2::GioIo? Or maybe Gphoto2, let me go and look for an old mail on this.

Andreas