Building and Using Bio-Formats with C++

Roger Leigh

Juin 2015
Institut Pasteur
Overview

Prerequisites
- Downloading
- Compiler and toolchain
- Package installation
- System and environment configuration

Building Bio-Formats
- Build overview
- Configuring and building
- Testing
- Installation

Using Bio-Formats

Future work
- In progress

Feedback
Java and C++

- bf-itk-pipe (pipe from C++ to JVM)
- JACE (wrap all Java classes, embed JVM)
Java and C++

- bf-itk-pipe (pipe from C++ to JVM)
- JACE (wrap all Java classes, embed JVM)
- Bio-Formats-C++ (native C++ implementation)
  - OME-TIFF Reference implementation
  - OME-XML model objects
  - Metadata store
  - Reading
  - Writing
Java and C++

- bf- itk-pipe (pipe from C++ to JVM)
- JACE (wrap all Java classes, embed JVM)
- Bio-Formats-C++ (native C++ implementation)
  - OME-TIFF Reference implementation
  - OME-XML model objects
  - Metadata store
  - Reading
  - Writing

- Initial uses:
  - Image acquisition writing OME-TIFF
  - Image analysis reading OME-TIFF
Source and documentation downloads

- (Download source)
- (Documentation)
- (Tutorial)
- (Doxygen API reference)
Default compilers

- FreeBSD: LLVM/clang++ or GCC/g++
- Linux: GCC/g++
- MacOS X: XCode (custom LLVM/clang++)
- Windows: Visual Studio or Visual Studio Express (MSVC/cl)
Default compilers

- FreeBSD: LLVM/clang++ or GCC/g++
- Linux: GCC/g++
- MacOS X: XCode (custom LLVM/clang++)
- Windows: Visual Studio or Visual Studio Express (MSVC/cl)

Note Visual Studio isn’t yet fully supported by the build and CI infrastructure but is in the pipeline
Package managers

- FreeBSD: Ports (e.g. pkg, portmaster)
- Linux: Distribution package manager (e.g. apt-get or yum)
- MacOS X: homebrew (brew)
- Windows: Yeah, right. You need to manually download all the tools and then compile all the libraries by hand for your specific version of Visual Studio. (Microsoft love to make development for their platform easy and painless. Not!)
Required packages

Libraries
- Boost
- HDF5
- PNG
- TIFF
- Xerces-C

Tools
- CMake
- Doxygen + Graphviz
- Git
- Graphicsmagick
- Python + genshi + sphinx
- \TeX Live
System configuration

- In general, none of the tools should require any configuration
- \LaTeX{} may require local font configuration to make the \TeX{} Gyre fonts available.
  - Linux and FreeBSD: Use the provided `fontconfig` template or create your own
  - MacOS X: Add to system using FontBook
  - Windows: May need adding to the system fonts if not found automatically
Environment configuration

- Primarily needed on Windows
- Rather than setting globally, make a batch file which can set up the environment.
- Activate a python virtualenv if needed
- Ensure that all tools are on the user PATH
  - cmake, doxygen, dot, git, python, sphinx, xelatex
- Set CMAKE_PREFIX_PATH if some libraries and tools are not on the default search path.
- Not all tools need to be on the default path; some will be discovered automatically by cmake
- No need to use a special Visual Studio shell when using cmake
cmake overview

CMakeLists.txt

- Compiler detection
- Program detection
- Library detection
- Feature tests
- Package options
- Rules for creating libraries programs, unit tests, installation

Cmake

Makefile (make)
or
project.sln and .vcxproj (Visual Studio msbuild)

Eclipse project
Sublime text project

...and many more build systems and IDES are supported
cmake features

- cmake is a generic cross-platform build system
- cmake generates build files for a large number of common build systems
- On FreeBSD, Linux and MacOS X, make Makefiles will be used
- On Windows with Visual Studio, msbuild .sln solution files will be used
- Eclipse, Sublime Text, Kate, Code::Blocks or several other IDEs or build systems may be used instead, if desired
Building Bio-Formats on Unix

Basic cmake usage

- Basic options
- Available generators

Build steps

- Configuring
- Building
- Testing
- Installing
Configure and build with `cmake`

Building from git or source release:
Configure the build:

% mkdir /tmp/bfbuid
% cd /tmp/bfbuid
% cmake /path/to/bioformats

Run the build with either of:

% make [VERBOSE=1]
% cmake --build .
Testing

Run the unit tests with any of:

% make test
% cmake --build . --target test
% ctest [-V]

Individual tests may be run by hand:

% ./bf-test cpp/test/ome-bioformats/pixelbuffer
Install the build with either of:

% make install [VERBOSE=1] [DESTDIR=/staging/path]
% cmake --build . --target install

By default, this will install into $CMAKE_INSTALL_PREFIX$ which will default to /usr/local. Use $DESTDIR$ to install into an alternative prefixed location, which is useful for testing and packaging for release.
Reading an OME-TIFF

```cpp
OMETIFFReader reader;

reader.setGroupFiles(true);
reader.setId(filename);
shared_ptr<MetadataStore> store =
    reader.getMetadataStore();

for (dimension_size_type series = 0U; series <
    reader.getSeriesCount(); ++series) {
    reader.setSeries(series);
    for (dimension_size_type plane = 0U; plane <
        reader.getPlaneCount(); ++plane) {
        VariantPixelBuffer pixels;
        reader.openBytes(plane, pixels);
    }
}
reader.close();
```
Writing an OME-TIFF

```cpp
shared_ptr<MetadataRetrieve> retrieve;
OMETIFFWriter writer;

writer.setMetadataRetrieve(retrieve);
writer.setInterleaved(interleaved);
writer.setId(filename);

for (dimension_size_type series = 0U; series < seriesCount; ++series) {
    writer.setSeries(series);
    for (dimension_size_type plane = 0U; plane < planeCount; ++plane) {
        VariantPixelBuffer pixels;
        writer.saveBytes(plane, pixels);
    }
}
writer.close();
```
Tools for testing

bf-test

- info—display image metadata
- view—Qt/OpenGL image viewer
In the pipeline

- Units
- 2015-01 data model
- bfconvert
- Windows support
- OME-XML model XSL transforms
- API improvements
Feedback

Which features do you need?

- Readers
- Writers
- Documentation
- Tools
- Support
- Integration with other software

Any feedback would be welcome and will help set and prioritise features and goals for future releases.
Acknowledgements

- OME Team, Dundee
  - Jason Swedlow
  - Jean-Marie Burel
  - Mark Carroll
  - Andrew Patterson
  - …and the rest of the team

- Micron, Oxford
  - Douglas Russell

- Glencoe Software
  - Melissa Linkert
  - Josh Moore