Derek Barnett – 23 March 2011

BamTools API Tutorial
Version 1.0

This document is intended to provide example uses, tips & tricks, etc. for using the BamTools API. It is **NOT** intended to provide full documentation of all classes and methods.

API Documentation
Link coming soon

Porting Guide (moving from BamTools 0.x to version 1.0)
https://github.com/pezmaster31/bamtools/wiki/BamTools-1x_PortingGuide.pdf

Table of Contents

Introduction
The API consists of a few main modules:

BamAlignment provides alignment data.
BamReader provides read access to BAM files.
BamWriter provides write access for generating BAM files.
BamMultiReader is a convenience class for working with a group of BAM files at once.
BamIndex is an interface hook for all index formats.*

* - This class straddles the line between a public API module and an internal implementation. Most client code will never need to use this class directly, but it does provide the capability for advanced clients to implement their own custom index schemes.

Add more introduction comments.

Derek Barnett – 23 March 2011

Programming with BamTools

First, be sure that BamTools has been built successfully. See the Building and Installing page for more information.

Next, import the API functionality as needed by including the corresponding headers. All BamTools classes and methods live in the “BamTools” namespace.

Below is a simple scenario for merging multiple BAM files, only keeping alignments that have a very high map quality. It should serve as introduction to some of the main classes, as well as the “feel” of the API:

```
#include “api/BamMultiReader.”
#include “api/BamWriter.h”
using namespace BamTools;

// at some point, start our merge operation

vector<string> inputFilenames;
string outputFilename;
// provide some input & output filenames

// attempt to open our BamMultiReader
BamMultiReader reader;
if ( !reader.Open(inputFilenames) ) {
    cerr << "Could not open input BAM files." << endl;
    return;
}

// retrieve 'metadata' from BAM files, these are required by BamWriter
const SamHeader header = reader.GetHeader();
const RefVector references = reader.GetReferenceData();

// attempt to open our BamWriter
BamWriter writer;
if ( !writer.Open(outputFilename, header, references) ) {
    cerr << "Could not open output BAM file" << endl;
    return;
}

// iterate through all alignments, only keeping ones with high map quality
BamAlignment al;
while ( reader.GetNextAlignmentCore(al) ) {
    if ( al.MapQuality >= 90 )
        writer.SaveAlignment(al);
}

// close the reader & writer
reader.Close();
writer.Close();

// merge is now complete, continue whatever we were doing
```

Add more usage examples & explanations.

Linking with BamTools

When it's time to build your application:

- Add (BAMTOOLS_ROOT)/include to your include paths (`-I`)
- Add (BAMTOOLS_ROOT)/lib to your library paths (-L)
- Link your app with '-lbamtools' ('l' as in Lima).

Depending on your platform and where you install the BamTools API library, you may also need to adjust how your app locates the shared library at runtime. For Windows users, this can be as simple as dropping the DLL in the same folder as your executable. For *nix users (using gcc at least), you can add the following to your app's CXXFLAGS:

```
-Wl,-rpath,$(BAMTOOLS_LIB_DIR)
```

where BAMTOOLS_LIB_DIR is the directory containing the libs. An alternative is to set your local LD_LIBRARY_PATH environment variable.

Another alternative is to use the newly provided static library libbamtools.a and resolve this issue at compile/link time, instead of runtime.

Note - For users that don't want to bother with using BamTools as a library: you are certainly free to just compile the API source code directly into your application, but be aware that the “internal” files are subject to change. Meaning that filenames, number of files, etc. are not fixed. You will also need to be sure to link with '-lz' for ZLIB functionality (linking with '-lbamtools' gives you this automatically).

Tips & Tricks

Add tips & tricks for performance boosts, etc.

Feel free to contact me if there is anything specific you would like to see added to this tutorial. Thanks and happy hacking!

Derek Barnett
Marth Lab, Boston College
derekwbarnett@gmail.com