About LAME

LAME is the one of most popular lossy audio codecs. It allows you to compress audio files to MPEG-1/MPEG-2 format, which is stored in MP3 files. LAME provides a wide range of compression bitrates: 8-320 kb/s and sampling rates: 8-48 kHz. Due to this codec popularity, one may be interested in implementing MP3 in native form through the Java Native Interface for an Android application. This article will go through the process of native compilation of the LAME library, implementing a simple wrapper for the encoder and creating a sample Android app that captures sound and encodes it.

Environment configuration

In order to use JNI in Android, the Android NDK is needed. It is a set of tools used to build Android app components in native code. It can be downloaded from the Android Developers site:

http://developer.android.com/sdk/ndk/index.html
After unpacking it, there will be an ndk-build script in the main directory. This is a script which automatically performs building processes with the NDK. It should be run from the root directory of the project to build or rebuild the native library. If you want to learn more about NDK itself, take a look at the documentation:

http://developer.android.com/sdk/ndk/overview.html

Preparing libraries

The LAME library can be downloaded from the following website:

http://lame.sourceforge.net/download.php
When the download is finished, an initial version (without a functional wrapper) of the shared library being created can be built. In order to do so, it is needed to create a directory called jni in the project's root. Then copy the libmp3lamedirectory from the downloaded archive to the created one. It is needed to copy all *.c and *.h files, so all others can be skipped. The main library header file named lame.h is also necessary. It is placed in the include directory in the downloaded archive. Copy it as the rest of files to the libmp3lame directory.
Some preparations are to be made yet. Since the standard compilation method with autoconf will not be used, some changes in header files need to be performed. First of all edit util.h and replace line:

extern ieee754_float32_t fast_log2(ieee754_float32_t x);
with used architecture 32-bit float implementation, which is simply float:
extern float fast_log2(float x);.
In set_get.h replace
#include <lame.h>
with
#include “lame.h”.

Compilation with NDK

Now create Makefile for NDK and include the LAME source files. The file should be named Android.mk and placed in the jni directory. It should look like this:

```
LOCAL_PATH := $(call my-dir)


                 include $(CLEAR_VARS)
```

```
                  LOCAL_MODULE            := libmp3lame
                  LOCAL_SRC_FILES         := \
                  ./libmp3lame/bitstream.c \
                  ./libmp3lame/encoder.c \
                  ./libmp3lame/fft.c \
                  ./libmp3lame/gain_analysis.c \
                  ./libmp3lame/id3tag.c \
                  ./libmp3lame/lame.c \
                  ./libmp3lame/mpglib_interface.c \
                  ./libmp3lame/newmdct.c \
                  ./libmp3lame/presets.c \
                  ./libmp3lame/psymodel.c \
                  ./libmp3lame/quantize.c \
                  ./libmp3lame/quantize_pvt.c \
                  ./libmp3lame/reservoir.c \
                  ./libmp3lame/set_get.c \
                  ./libmp3lame/tables.c \
                  ./libmp3lame/takehiro.c \
                  ./libmp3lame/util.c \
                  ./libmp3lame/vbrquantize.c \
                  ./libmp3lame/VbrTag.c \
                  ./libmp3lame/version.c

                  LOCAL_LDLIBS := -llog

                  include $(BUILD_SHARED_LIBRARY)
```

The shared library can now be built. Go to jni directory and call ndk-build script from there. A proper compilation will give an output like this:

```
[…]
                  SharedLibrary  : libmp3lame.so
                  Install        : libmp3lame.so => libs/armeabi/libmp3lame.so

                  There should be new folders in the project root: libs and obj.
```

Creating a wrapper

Now is the point at which a wrapper, which will be the interface for the future Android app to use with the recently compiled LAME library, is to be created. Create wrapper.c file in jni directory and include it in Android.mk file in LOCAL_SRC_FILES variable:

```
LOCAL_SRC_FILES :=  \
                  ./libmp3lame/bitstream.c \
                   [...]
                  ./libmp3lame/version.c \
                  ./wrapper.c
```

The wrapper should be adapted to particular needs. The sample implementation will provide three methods:

- initEncoder, which will prepare the encoder to work with given channels number, sampling rate, bit rate, mode and quality
- destroyEncoder, which will uninitialize encoder
- encodeFile, which will encode source raw file to the target MP3 file; method will receive paths to both of these files as parameters

If you don't have any experience with LAME programming, you should first refer to the API file placed in the downloaded archive and the lame.h file included into the library, which describes most of the LAME API methods.

You may also want to get familiar with JNI specific functions:
http://docs.oracle.com/javase/1.5.0/docs/guide/jni/spec/functions.html
When the wrapper is finished, rebuild the shared library with ndk-build. The output should look like this:

```
Compile thumb  : mp3lame <= wrapper.c
                         SharedLibrary  : libmp3lame.so
                         Install        : libmp3lame.so => libs/armeabi/libmp3lame.so
```

Creating an Android application

Since a functional wrapper for the encoder has been made, it can be used in Java code. First, let the app record audio and access the SD card to store raw and encoded files by adding the required Android permissions: RECORD_AUDIO and WRITE_EXTERNAL_STORAGE.
The next step is to load the native library. Do it in a static block, so it is performed only once. Then declare the native methods:

```
static {
                     System.loadLibrary("mp3lame");
                 }
                 private native void initEncoder(int numChannels, int sampleRate, int
bitRate, int mode, int quality);
                 private native void destroyEncoder();
                 private native int encodeFile(String sourcePath, String targetPath);
```

Now they can be used as normal. The sample app delivered with this article has a simple layout with one button, which starts or stops recording. The recorder and encoder are initialized at the beginning. After pressing the button to start, the app will begin capturing the sound and write it to the buffered output. When decided to stop, sound capturing will be interrupted and the captured data will be passed on to the encoder. When encoding is finished, the user will be notified with the encoded file name. When the app finishes, the recorder and encoder are released.

How to run the sample app?

The sample project delivered with this article does not contain any third-party code. You will need to prepare the LAME library as it is described in the Preparing libraries paragraph and then compile them with the ndk-build script.
Ref:http://developer.samsung.com/android/technical-docs/Porting-and-using-LAME-MP3-on-Android-with-JNI