Directly reading the MEF format

Eric Kim

August 12, 2010

Abstract

The mex file decomp_mef.c uses the start and end indices specified by the user to scan through MEF data and decompress the desired data samples. This document focuses on the process of compiling and implementing decomp_mef.c.

1 mex basics

- To compile a mex file in MATLAB, use the mex command:

  ```
  mex filename.c
  ```

- To choose the C compiler for compilation:

  ```
  mex -setup
  ```

2 Linked files

decomp_mef.c should be compiled with the following files: RED_decode.c endian_functions.c mef_lib.c AES_encryption.c crc_32.c and RED_encode.c. To do so in MATLAB, simply include the linked files after decomp_mef.c when making the mex file:

```
mex decomp_mef.c RED_decode.c endian_functions.c mef_lib.c
AES_encryption.c crc_32.c RED_encode.c
```

3 Operating system compatibility

decomp_mef.c was oringinally written for UNIX-based operating systems and should compile cleanly on most unix platforms with small differences in detail. Compiling on windows is more challenging, though doable.

3.1 Ubuntu

Comments

There is a good chance the C compilers installed on ubuntu will only support C89 and C90. This presents a compatibility issue because inline comments with // syntax were introduced in C99. If compiling results in:

```
error:  expected identifier or ( before / token
```

this indicates that a double-slash comment has been used. All inline comments with // should be replaced with /* */ style comments.

GCC version

Many of the later versions of gcc do not support mex files:

```
Warning:  You are using gcc version "4.4.3-4ubuntu5)".
The earliest gcc version supported with mex is "4.1".  The
latest version tested for use with mex is "4.2".  To download
a different version of gcc, visit http://gcc.gnu.org
```

To eliminate this warning, first install the correct version of gcc:

```
sudo apt-get install gcc-4.1-multilib
```

Next, change the gcc symbolic link to make sure it refers to the correct version of gcc:

```
sudo mv /usr/bin/gcc /usr/bin/gcc_mybackup
sudo ln -s /usr/bin/gcc-4.1 /usr/bin/gcc
```

If you want to compile something else using the later version of gcc, you can restore the symbolic link with:

```
sudo mv /usr/bin/gcc_mybackup /usr/bin/gcc
```

Object Files

Compiling decomp_mef.c works best on ubuntu if the linked files are made into object files first. This can be done in the terminal with:

```
cc -c filename.c
```

After the object files have been created, make the mex file in MATLAB using the object files instead:

```
mex decomp_mef.c RED_decode.o endian_functions.o mef_lib.o
```

```
AES_encryption.o crc_32.o RED_encode.o
```

3.2 Mac

Compiling on a mac should not present errors:

```
mex decomp_mef.c RED_decode.c endian_functions.c mef_lib.c
AES_encryption.c crc_32.c RED_encode.c
```

should create the appropriate mex file.

3.3 Windows

Header Files

Because decomp_mef.c was originally written for unix-based operating systems, it includes two header files (pthread.h and sched.h) that are a part of the standard C library on unix-based platforms, but are not present on windows. Thus, we must manually download windows compatible versions of the missing header files from the GNU C library. These can quickly be found by googling "GNU C library *header file*."

fseeko

fseeko is only available on unix but appears several times in decomp_mef.c. Replace every instance of fseeko with fseek. Also, fseeko takes in an argument of type (off_t); this is not recognized by windows. Thus, replace every instance of (off_t) with (long). Both off_t and long datatypes are 64 bit datatypes, which reduces the potential loss of data. Fseeko and fseek are similar enough that the decompression should still run smoothly.

Unneccessary Functions

A few functions in the linked files will cause problems when attempting to compile decomp_mef.c on windows. However, these functions are used only when compressing data in the MEF format and are not necessary to run decomp_mef.c. To eliminate these errors, we don't include RED_encode.c (which is only completely necessary during compression), and comment out the following functions in mef_lib.c: build_mef_header_block, generate_unique_ID, set_hdr_unique_ID, set_block_hdr_unique_ID, set_session_unique_ID and write_mef. NOTE: Compiling on unix based operating systems can be done in this way as well, though it is unnecessary because these functions do not cause errors on unix.

fopen/fread

Unix systems treat text files and binary files the same. However, windows treats these differently and often fails to correctly read text files. In order to eliminate this problem, open the MEF data as a binary file with the "rb" mode instead of the "r" mode.

Final Compilation

With the above erorrs eliminated, the mex file can be compiled without creating object files:

```
mex decomp_mef.c RED_decode.c endian_functions.c mef_lib.c
AES_encryption.c crc_32.c
```

4 Endianess

Currently, decomp_mef.c will only run on little-endian machines. In practice, this should not present a significant problem given the limited number of machines using big-endian datatypes. To run decomp_mef.c on big-endian machines would require extensive changes in the coding.