1 C++ Coding Challenge

1.1 Introduction

The castLabs Android SDK, iOS SDK and DASH Everywhere products are video player SDKs that runs on all relevant platforms. Some of the noteworthy features are multiple language and subtitle support, cross-platform compatibility and support of multiple DRM schemes for content protection.

The content (e.g. a video or audio track) is given in a container format and streamed to the player via the DASH or Smooth Streaming protocol.

1.2 The MPEG-4 Part 12 ISO Base Media File Format

For both DASH and Smooth Streaming content, the files are based on the MPEG-4 Part 12 ISO Base Media File Format. Details about the format can be found at `https://en.wikipedia.org/wiki/ISO_base_media_file_format`, but for this exercise, it is sufficient to know the general layout of such a file:

Files conforming to the ISO base media file format are formed as a series of objects, called "boxes". All data is contained in boxes and there is no other data within the file. This includes any initial signature required by the specific file format. The "box" is object-oriented building block defined by a unique type identifier and length. It was called "atom" in some specifications (e.g. the first definition of MP4 file format).

To view the structure and layout of such a file, we recommend using ISO Viewer, which can be downloaded at:

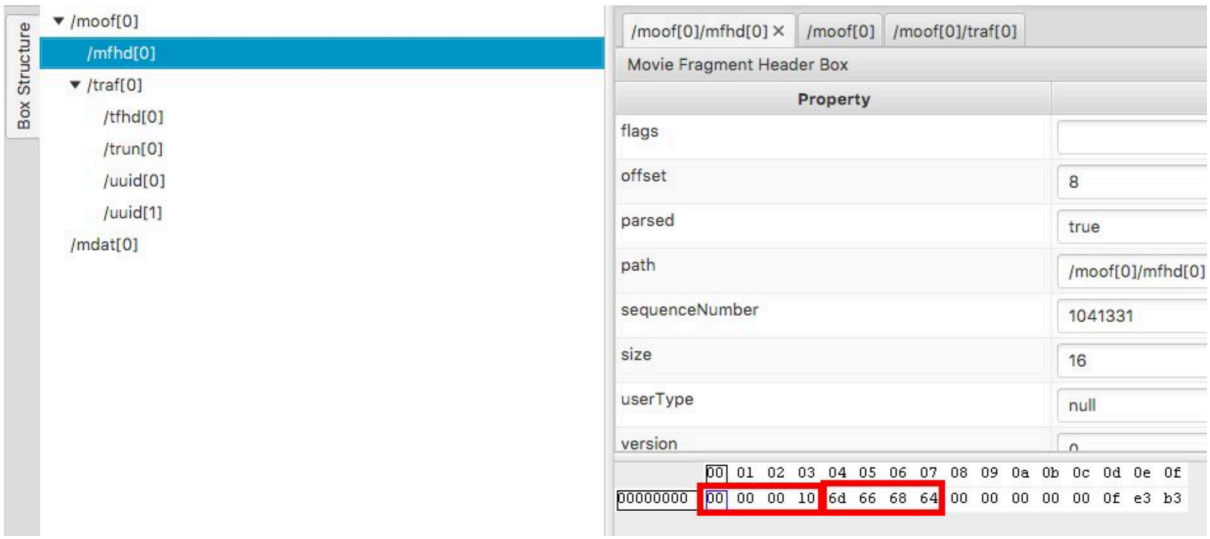
`https://github.com/sannies/isoviewer`

1.3 Sample File

For this exercise, the following sample MP4 file will be used:

`http://demo.castlabs.com/tmp/text0.mp4`

It's layout can be seen using ISO Viewer:



The file contains multiple, nested boxes, as seen in the tree-structure of ISO Viewer. The file contains one *MOOF* box (which itself has two sub boxes) and one *MDAT* box:

- MOOF
    - MFHD
    - TRAF
        - TFHD
        - TRUN
        - UUID
        - UUID
- MDAT

As seen, each box can contain other boxes (sub boxes) or actual data. The basic layout that all kind of boxes have in common is this:

The first four bytes (bytes 0-3) specify the size (or length) of the box. Bytes 4-7 specify the type of the box. Without knowing any more details about the different types of boxes, this knowledge is enough to read the basic structure of such a file.

1.4 Example

Let's take a look at the *MFHD* box from the sample file. The first four bytes are:

```
00 00 00 10
```

which indicates a box size of 16 bytes. Bytes 4-7 are

```
6D 66 68 64
```

which tells us the box type *MFHD*.

The box size includes the 8 bytes for the size and the type, so in this example, there are 8 remaining bytes for actual box data.

1.5 Exercise

In this exercise, a C++ application should be written that prints the layout of the sample file and extracts the content of the *MDAT* box. The application should work in either Linux/OS X/Windows and you are free to use whatever libraries you see fit, i.e. Boost/Qt/POCO.

It should provide the following:

1. Read the sample file from

```
http://demo.castlabs.com/tmp/text0.mp4
```

2. Iterate through the file and print the size and type of each box found to the console. The following assumptions can be made:

a. A box of type *MOOF* only contains other boxes

b. A box of type *TRAF* only contains other boxes

c. All other boxes don't contain other boxes.

3. If the box of type *MDAT* is found, extract and print the content of that box. It can be assumed that the content is a UTF-8 encoded XML string.

Sample output of such an application:

```
2015-11-30 18:11:55.644 Successfully loaded file http://demo.castlabs.com/tmp/text0.mp4
2015-11-30 18:11:55.650 Found box of type moof and size 181
2015-11-30 18:11:55.650 Found box of type mfhd and size 16
2015-11-30 18:11:55.650 Found box of type traf and size 157
2015-11-30 18:11:55.651 Found box of type tfhd and size 24
2015-11-30 18:11:55.651 Found box of type trun and size 20
2015-11-30 18:11:55.651 Found box of type uuid and size 44
2015-11-30 18:11:55.651 Found box of type uuid and size 61
2015-11-30 18:11:55.651 Found box of type mdat and size 17908
2015-11-30 18:11:55.652 Content of mdat box is: <?xml version="1.0" encoding="UTF-8"?><tt xml:lang="
xmlns:tts="http://www.w3.org/ns/ttml#styling" xmlns:smpte="http://www.smpte-ra.org/schemas/2052-1/20
<smpte:information smpte:mode="Enhanced" />
<styling>
<style xml:id="emb" tts:fontSize="4.1%" tts:fontFamily="monospaceSansSerif" />
<style xml:id="ttx" tts:fontSize="3.21%" tts:fontFamily="monospaceSansSerif"/>
<style xml:id="backgroundStyle" tts:fontFamily="proportionalSansSerif" tts:fontSize="18px" tts:textA
```

Bonus 1: Which problem can occur if the content of the *MDAT* box is very large?

Bonus 2: The MDAT box contains base64-encoded images. Extract those images to files.

1.6 Delivery

Your result should be delivered in a public github repository that also exposes the development process.

There is no deadline to complete the exercise, so it is up to you after how many days you would return your solution.