Parser Coding Challenge

Languages: C, C++ or Python

The MPEG4 Part 12 ISO Base Media File Format

This exercise uses the MPEG4 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, you can use:

https://gpac.github.io/mp4box.js/test/filereader.html

Sample File

https://drive.google.com/file/d/1io0pfOXPoAEqYIOgQ_rS1EiZVr-hQiT8/view?usp=drive_link

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 (it's a 4 Byte Big Endian Integer).
- Bytes 4 - 7 specify the type of the box (each byte can read as an ASCII character).

Without knowing any more details about the different types of boxes, this knowledge is enough to read the basic structure of such a file.

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.

Exercise

In this exercise, a C, C++ or Python application should be written that prints the layout of the sample file (showing how boxes are nested, e.g. with indentation) and extracts the content of the MDAT box. The application should work in either Linux, macOS or Windows and you are free to use whatever libraries you see fit, however do not use a library that is already doing the parsing itself as that’s the point of this exercise.

It should provide the following:

1. Read the sample file from disk
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 UTF8 encoded XML string.

Sample output of such an application:

```
Box ID: moof of size 181
    Box ID: mfhd of size 16
    Box ID: traf of size 157
       Box ID: tfhd of size 24
       Box ID: trun of size 20
       Box ID: uuid of size 44
       Box ID: uuid of size 61
Box ID: mdat of size 17908
Mdat content:
<?xml        version="1.0"        encoding="UTF-8"?><tt        xml:lang="spa"
xmlns="http://www.w3.org/ns/ttml"
xmlns:tts="http://www.w3.org/ns/ttml#styling"
xmlns:smpte="http://www.smpte-ra.org/schemas/2052-1/2010/smpte-tt" ><head>
<smpte:information smpte:mode="Enhanced" />
<styling>
Etc. etc.
```

Bonus 1: Which problem can occur if a box has a very large size?

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

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.