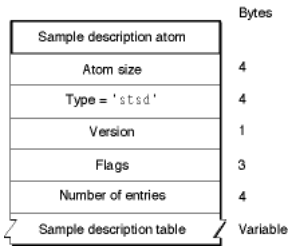
c实现mp4解封装

Created	@August 20, 2022 5:29 PM
Status	Done



前序

最近为了更加深入了解音视频demux这块的功能，准备着手写个demuxer，提取视频流。

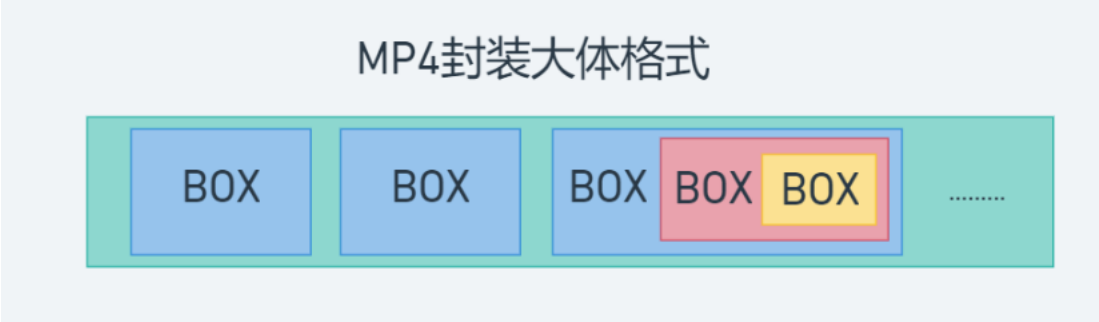
MP4简介

MP4的定义

MP4是一种常用的视音频流封装格式，按照指定的协议来存放媒体数据；因为mp4是基于苹果QuickTime文件格式，所以与mov有很多相同之处，在苹果开发者平台可以看到详细的有关封装文档(https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html#//apple_ref/doc/uid/TP40000939-CH204-25615)

MP4的封装格式

- MP4格式预览—mp4是由多个box嵌套组成的



- MP4主要的顶部box

  ftyp box：描述MP4锁遵循的规范和版本

  mdat box ：存放媒体数据

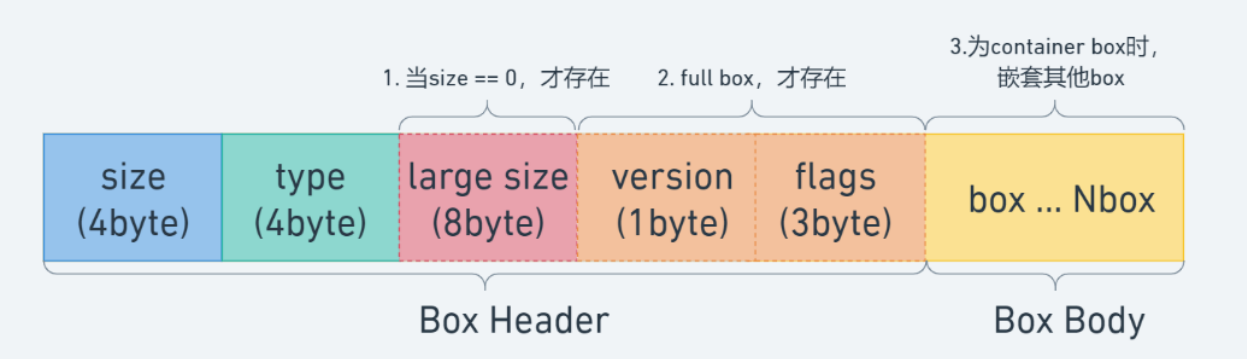
  moov box：存放媒体参数(pps、sps等)相关信息和用于索引媒体数据存储位置的信息

- MP4常用box

Type						Mandatory	Description
ftyp						*	file type and compatibility
moov						*	container for all the metadata
	mvhd					*	movie header, overall declarations
	trak					*	container for an individual track or stream
		mdia				*	container for the media information in a track
			mdhd			*	media header, overall information about the media
			hdlr			*	handler, declares the media (handler) type
			minf			*	media information container
				vmhd			video media header, overall information (video track only)
				smhd			sound media header, overall information (sound track only)
				stbl		*	sample table box, container for the time/space map
					stsd	*	sample descriptions (codec types, initialization etc.)
					stts	*	(decoding) time-to-sample
					ctts		(composition) time to sample
					stsc	*	sample-to-chunk, partial data-offset information
					stsz		sample sizes (framing)
					stz2		compact sample sizes (framing)
					stco	*	chunk offset, partial data-offset information
					stss		sync sample table
mdat							media data container

Box类型详解

Box格式

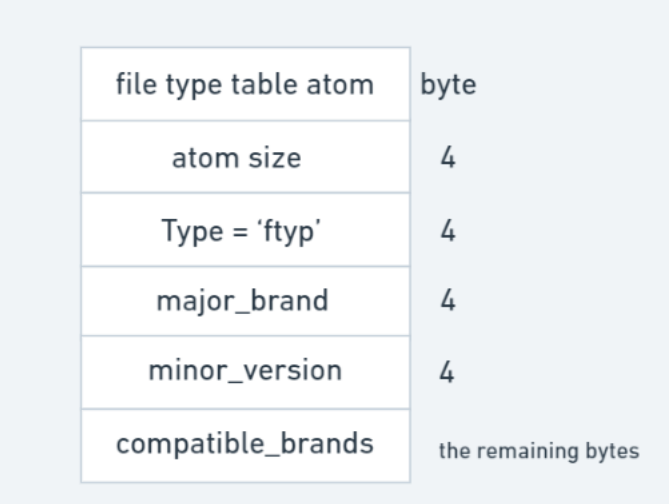


1. size字段为整个box的大小，包括box header和box body

2. type为box的类型，通常为四字节的字符串，例如ftyp
3. 当size == 0时，box的大小为large size
4. 当box为full box时，存在version和flags字段，具体含义因box不同而不同
5. 若box没有嵌套其他box，例如ftyp box，则box body部分根据具体规范解析相应字段；若box为container box，则box body部分嵌套其它box，还需一步步解套获取最终的数据

ftyp box

- 字段分布图

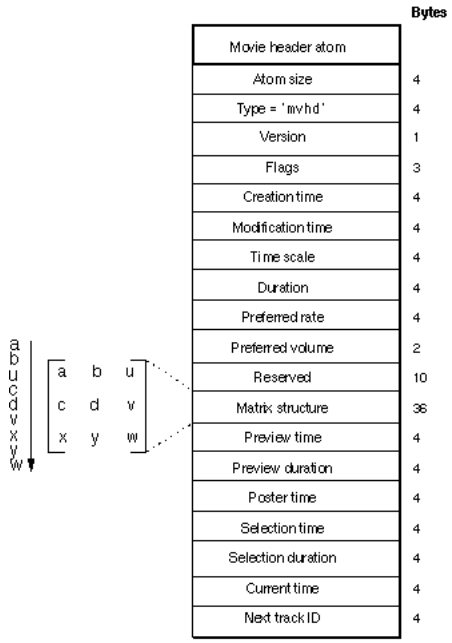


file type table atom	byte
atom size	4
Type = 'ftyp'	4
major_brand	4
minor_version	4
compatible_brands	the remaining bytes

- 字段解析
- major_brand：比如常见的 isom、mp41、mp42、avc1、qt等。它表示"最好"基于哪种格式来解析当前的文件。举例，major_brand 是 A，compatible_brands 是 A1，当解码器同时支持 A、A1 规范时，最好使用A规范来解码当前媒体文件，如果不支持A规范，但支持A1规范，那么，可以使用A1规范来解码；
- minor_version：提供 major_brand 的说明信息，比如版本号，不得用来判断媒体文件是否符合某个标准/规范；
- compatible_brands：文件兼容的brand列表。比如 mp41 的兼容 brand 为 isom。通过兼容列表里的 brand 规范，可以将文件 部分（或全部）解码出来；

mvhd box

- 字段分布图



- 字段解析
  - version ：一字节用于指定mvhd的版本
  - flags：3字节，预留
  - Create time：媒体创建时间，与UTC时间不同的是，此时间是从1904年1月1日0:0:0开始计算的，而utc是从1970年1月1日0:0:0开始计算，故Create time须要减去时间差换算成utc时间

    ```
    // 注：66年时间差不是66*365*24*3600来计算
    creation_time_utc = creation_time - (66年时间差) = creation_time - 2082844800
    ```

  - Modification time：媒体最后被修改的时间，计算方式同Create time
  - Timescale：一秒包含的时间单位（整数）。举个例子，如果timescale等于1000，那么，一秒包含1000个时间单位（后面track等的时间，都要用这个来换算，比如track的duration为10,000，那么，track的实际时长为10,000/1000=10s）；
  - Duration：影片时长（整数），根据文件中的track的信息推导出来，等于时间最长的track的duration；
  - Preferred rate：推荐的播放速率，32位整数，高16位、低16位分别代表整数部分、小数部分（[16.16]），举例 0x0001 0000 代表1.0，正常播放速度；
  - Preferred volume：播放音量，16位整数，高8位、低8位分别代表整数部分、小数部分（[8.8]），举例 0x01 00 表示 1.0，即最大音量；
  - Matrix struct：视频的转换矩阵，详情看

    Basic Data Types

    This chapter describes a number of common data types that are used in QuickTime files. Some elements of a QuickTime file may be associated with a particular spoken language. To indicate the language associated with a particular object, the QuickTime file format uses either language codes from the

    https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap4/qtff4.html#//apple_ref/doc/uid/TP40000939-CH206-18737

  - Next_track_ID：32位整数，非0，一般可以忽略不计。当要添加一个新的track到这个影片时，可以使用的track id，必须比当前已经使用的track id要大。也就是说，添加新的track时，需要遍历所有track，确认可用的track id；

tkhd box

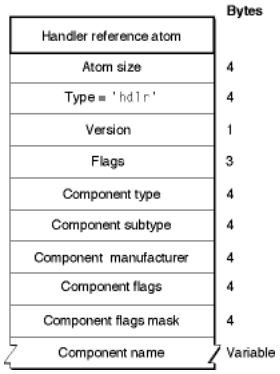
- 字段分布图

Track header atom	Bytes
Atom size	4
Type = 'tkhd'	4
Version	1
Flags	3
Creation time	4
Modification time	4
Track ID	4
Reserved	4
Duration	4
Reserved	8
Layer	2
Alternate group	2
Volume	2
Reserved	2
Matrix structure	36
Track width	4
Track height	4

- 字段解析
  - version：tkhd box的版本；
  - flags：按位或操作获得，默认值是7（0x000001 | 0x000002 | 0x000004），表示这个track是启用的、用于播放的 且 用于预览的。
    - Track_enabled：值为0x000001，表示这个track是启用的，当值为0x000000，表示这个track没有启用；
    - Track_in_movie：值为0x000002，表示当前track在播放时会用到；
    - Track_in_preview：值为0x000004，表示当前track用于预览模式；
  - Creation time：当前track的创建时间；
  - Modification time：当前track的最近修改时间；
  - Track ID：当前track的唯一标识，不能为0，不能重复；
  - Duration：当前track的完整时长（需要除以timescale得到具体秒数）；
  - Layer：视频轨道的叠加顺序，数字越小越靠近观看者，比如1比2靠上，0比1靠上；
  - Alternate_group：当前track的分组ID，alternate_group值相同的track在同一个分组里面。同个分组里的track，同一时间只能有一个track处于播放状态。当alternate_group为0时，表示当前track没有跟其他track处于同个分组。一个分组里面，也可以只有一个track；
  - Volume：audio track的音量，介于0.0~1.0之间；
  - Matrix structure：视频的变换矩阵；
  - Track width：视频的宽
  - Track height：视频的高

hdlr box

- 字段分布图



- 字段解析
  - Version：hdlr box的版本
  - Flags：置0
  - Component type：四字节子串定义handler的类型；此字段只有两种值合法：'mhlr'(media handlers)和'dhlr'(data handlers)
  - Component subtype：针对Component type进行细分类型，例如'vide'定义为视频数据，'soun'定义为音频数据
  - Component manufacturer：保留，置0
  - Component flags：保留，置0
  - Component flags mask：保留，置0
  - Component name：子串指定Component 的名字，可能为空

mdat box

- 数据结构分布图

注意：取到的frame前四个字节为frame数据的长度字节，须要偏移去掉

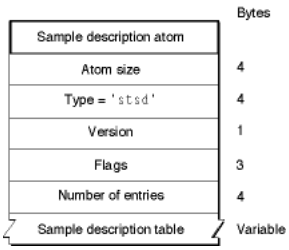
stbl box

主要存放了媒体参数(pps、sps、vps等)相关信息和用于解析mdat中视音频数据的关键信息

- stsd：给出视音频的相关参数信息，有高宽、音量、位深度和每个sample多少个frame
- stco：thunk在文件中的偏移
- stsc：每个thunk中包含几个sample
- stsz：每个sample的size（单位是字节）
- stts：每个sample的时长
- stss：哪些sample是关键帧

stsd box

- 字段分布图



- 字段解析
  - Version：stsd box的版本
  - Flags：置0
  - Number of entries：Sample description table的个数
  - Sample description table：以视频为例，此时Sample description table字段中为若干个视频编码相关的box，例如avc1 box
    - avc1 box
      - 字段分布图

file type table atom	byte
atom size	4
Type = ‘avc1’	4
reserved	6
Data reference index	2
version	2
revidion_level	2
vendor	4
temporal_quality	4
spatial_quality	4
width	2
height	2
horizonta_resolution	4
vertical_resolution	4
data_size	4
frame_count	2
compressor_name	32
depth	2
color_table_id	2

    - 字段解析

      这个在https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html的“**Sample Description Atoms**”一节可以研究下

- avcC box(包含了视频关键参数，在ISO/IEC 14496-15中定义)
    - 字段分布图

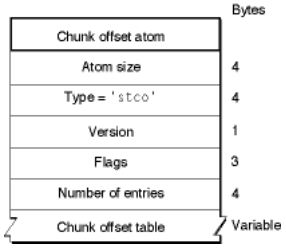
file type table atom	byte
atom size	4
Type = 'avcC'	4
configuration_version	1
avc_profile_indication	1
profile_compatibility	1
avc_level_indication	1
length_size_minusOne	1
num_of_sps	1
sps_length	2
sps_nal_unit	sps_length
num_of_pps	1
pps_length	2
pps_nal_unit	pps_length

- 字段解析
  - num_of_sps：sps的个数
  - sps_length：sps的长度
  - sps_nal_unit：长度为sps_length的sps
  - num_of_pps：pps的个数
  - pps_length：pps的长度
  - pps_nal_unit：长度为pps_length的pps

  其他字段可以自行在ISO/IEC 14496-15中查到

stco box

- 字段分布图

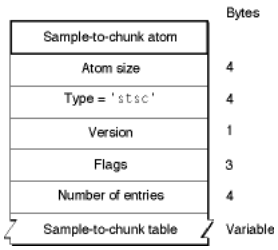


- 字段解析
  - Version：stco box的版本

- Flags：置0
- Number of entries：chunk的个数
- Chunk offset table：每个chunk在整个视频文件的偏移值，每个值的长度为4字节

stsc box

- 字段分布图



Sample-to-chunk atom	Bytes
Atom size	4
Type = 'stsc'	4
Version	1
Flags	3
Number of entries	4
Sample-to-chunk table	Variable

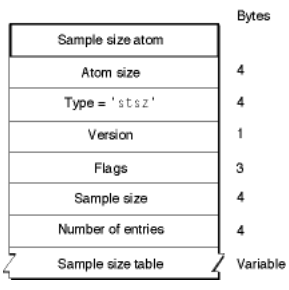
- 字段解析
  - Version：stsc box的版本
  - Flags：置0
  - Number of entries：”Sample-to-chunk table”的条数
  - Sample-to-chunk table：
    - First chunk：chunk的索引
    - Samples per chunk：从’First chunk’开始，每个chunk中sample的个数
    - Sample description ID：stsd box中‘Sample description table’的下标
- Sample-to-chunk table示意图

First chunk	Samples per chunk	Sample description ID
1	3	23
3	1	23
5	1	24

  - chunk1-chunk2：每个chunk中有3个sample，并且Sample description ID为23
  - chunk3-chunk4：每个chunk中有1个sample，并且Sample description ID为23
  - chunk5：每个chunk中有3个sample，并且Sample description ID为24

stsz box

- 字段分布图

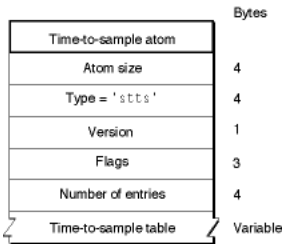


Sample size atom	Bytes
Atom size	4
Type = 'stsz'	4
Version	1
Flags	3
Sample size	4
Number of entries	4
Sample size table	Variable

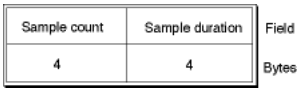
- 字段解析
  - Version：**stsz** box的版本
  - Flags：置0
  - Sample size：为0则表示所有sample的大小不一定一样，不为0则表示所有sample的大小一样
  - Number of entries：”Sample size table”的条数
  - Sample size table：每个sample的size，每个sample size的长度为4字节

stts box

- 字段分布图



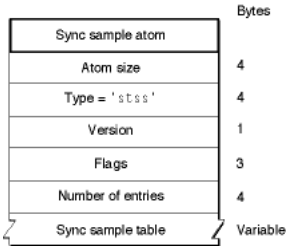
- 字段解析
  - Version：**stts** box的版本
  - Flags：置0
  - Number of entries：“Time-to-sample table”数组的长度
  - Time-to-sample table：
    - Sample count：具有相同“Sample duration”的个数
    - Sample duration：sample的时长（以timescale为计量)
- Time-to-sample table：示意图



sample1 - sample4的sample duration是4

stss box

- 字段分布图



- 字段解析
  - Version：**stts** box的版本
  - Flags：置0

- Number of entries：“Sync sample table”的条数
- Sync sample table：关键帧对应的sample index

demuxer demo的实现(视频数据部分)

1. 获取sps pps参数
   a. 解析stsd box，其中contain avc1 box和avcC box(此步骤详解见上文)
   b. 解析avcC box可以获取到sps和pps

      以下为ISO/IEC 14496-15中解析avcC的伪代码

```
aligned(8) class AVCDecoderConfigurationRecord {
   unsigned int(8) configurationVersion = 1;
   unsigned int(8) AVCProfileIndication;
   unsigned int(8) profile_compatibility;
   unsigned int(8) AVCLevelIndication;
   bit(6) reserved = ‘111111’b;
   unsigned int(2) lengthSizeMinusOne;
   bit(3) reserved = ‘111’b;
   unsigned int(5) numOfSequenceParameterSets;
   for (i=0; i< numOfSequenceParameterSets; i++) {
     unsigned int(16) sequenceParameterSetLength ;
     bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;
   }
   unsigned int(8) numOfPictureParameterSets;
   for (i=0; i< numOfPictureParameterSets; i++) {
     unsigned int(16) pictureParameterSetLength;
     bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;
 }

 if( profile_idc == 100 || profile_idc == 110 ||
   profile_idc == 122 || profile_idc == 144 )
   {
     bit(6) reserved = ‘111111’b;
     unsigned int(2) chroma_format;
     bit(5) reserved = ‘11111’b;
     unsigned int(3) bit_depth_luma_minus8;
     bit(5) reserved = ‘11111’b;
     unsigned int(3) bit_depth_chroma_minus8;
     unsigned int(8) numOfSequenceParameterSetExt;
     for (i=0; i< numOfSequenceParameterSetExt; i++) {
       unsigned int(16) sequenceParameterSetExtLength;
       bit(8*sequenceParameterSetExtLength) sequenceParameterSetExtNALUnit;
     }
   }
}
```

2. 获取关键帧位置

   解析stss box可以知道哪一个sample中包含关键帧
3. 获取chunk位置

   解析stco box可以获取到每个chunk在视频文件中的索引
4. 获取每个chunk中sample个数

   解析stsc box可以获取到每个chunk包含多少个sample
5. 获取sample大小

   解析stsz box可以获取到每个sample的大小
6. 获取frame位置(demo视频文件一个sample只包含一个frame，所以sample的位置和大小就是frame的位置和大小)
   a. 根据stsd解析到每个sample中有多少个frame
   b. 然后再根据trunk的位置和sample的大小来定位frame起始地址
   c. mdat中frame的数据格式为：| 4字节数据长度 | frame数据|，所以根据字节长度读取相应个数frame
7. 获取到一帧数据后

a. 判断当前frame为I帧，则添加写入(start_code+sps) + (start_code+pps) + (start_code + frame数据)到输出文件

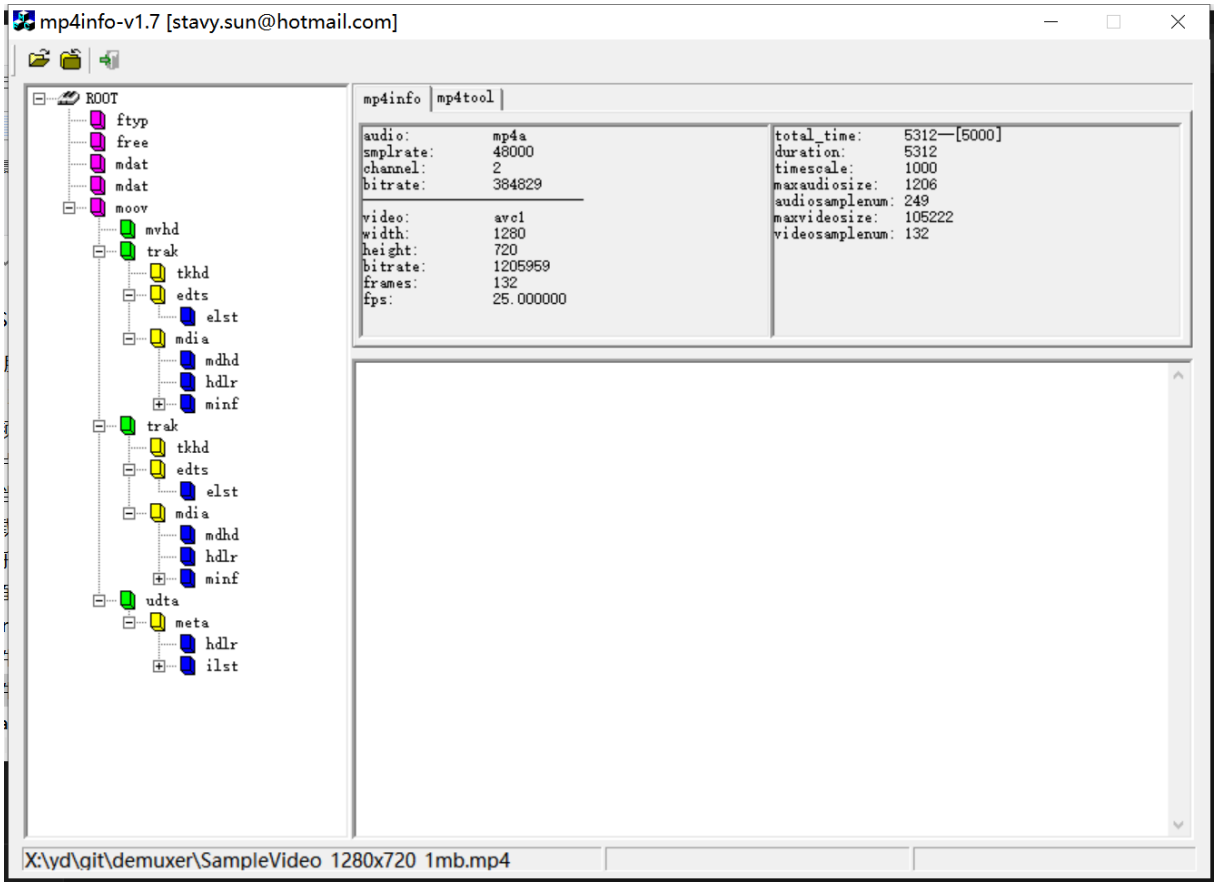
b. 判断当前frame不为I帧，则写入(start_code + frame数据)到输出文件

8. 保存成h264文件，可使用ffplay和potplay播放

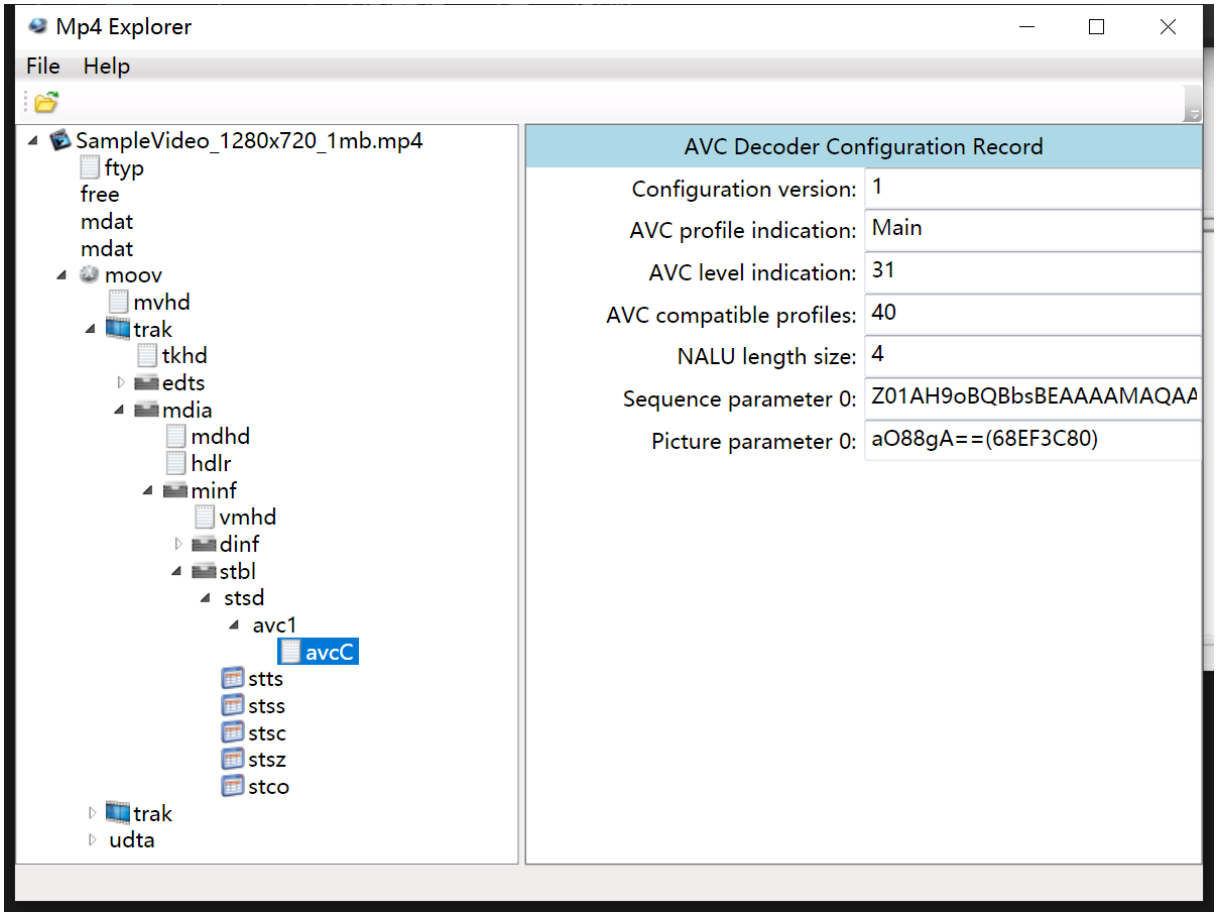
注意：有些非字串的字段为大端字节序，须要转换

工具介绍

1. mp4info—可以看到相关box的字节信息，但发现对avcC的解析漏掉了几个字节



2. mp4 exploer—可以更加直观的看到视音频数据信息



参考

1. https://zhuanlan.zhihu.com/p/333765990
2. https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html#//apple_ref/doc/uid/TP40000CH204-25691
3. ISO/IEC 14496-15

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ce00f666-85a9-45e6-a68c-64bc307e7e34/Untitled.pdf