## 一、mp4概述

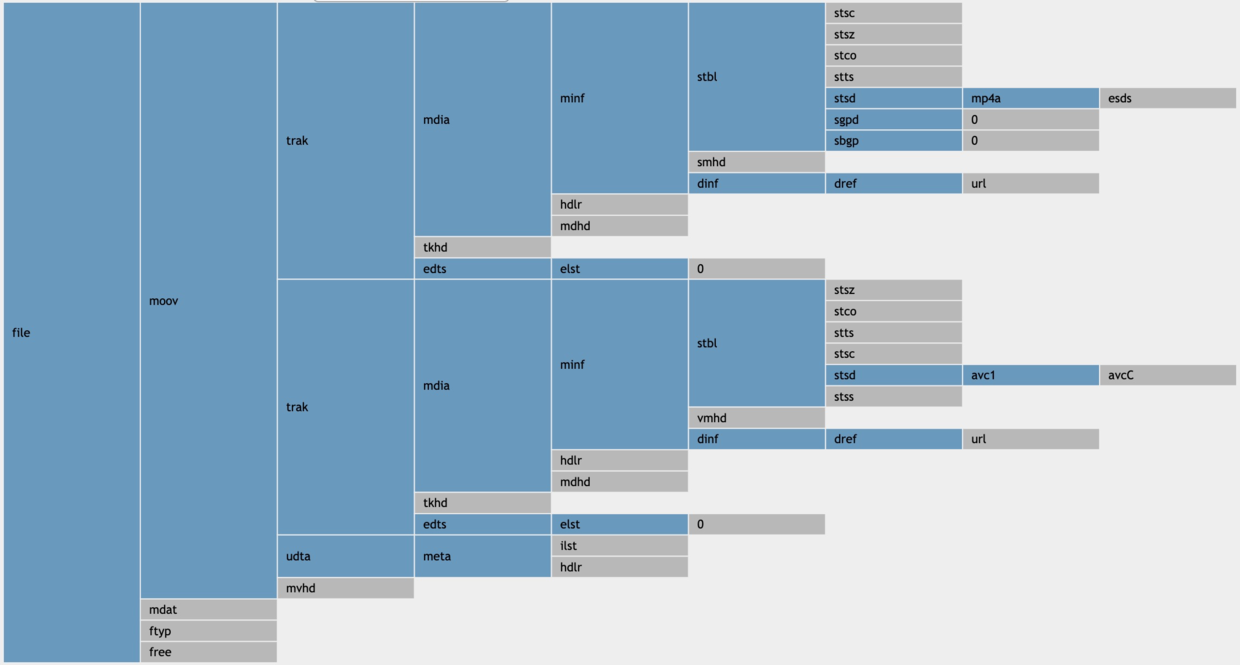
MP4由国际标准化组织（ISO）和国际电工委员会（IEC）下属的“动态图像专家组”（Moving Picture Experts Group，即MPEG）制定。MPEG-4格式的主要用途在于网上流、光盘、语音发送（视频电话），以及电视广播，是一种常见的多媒体封装格式。

MP4 文件格式又被称为 MPEG-4 Part 14，出自 MPEG-4 标准第 14 部分 。它是一种多媒体格式容器，广泛用于包装视频和音频数据流、海报、字幕和元数据等。

目前流行的视频编码格式 AVC/H264 定义在 MPEG-4 Part 10。

MP4 文件格式基于 Apple 公司的 QuickTime 格式。

## 二、MP4文件物理结构（常见）



1. MP4 文件由 box 组成，每个 box 分为 Header 和 Data。其中 Header 部分包含了 box 的类型和大小，Data 包含了子 box 或者数据，box 可以嵌套子 box。
2. 一个MP4文件首先会有且只有一个“ftyp”类型的box，作为MP4格式的标志并包含关于文件的一些信息，描述文件遵从的MP4规范与版本。

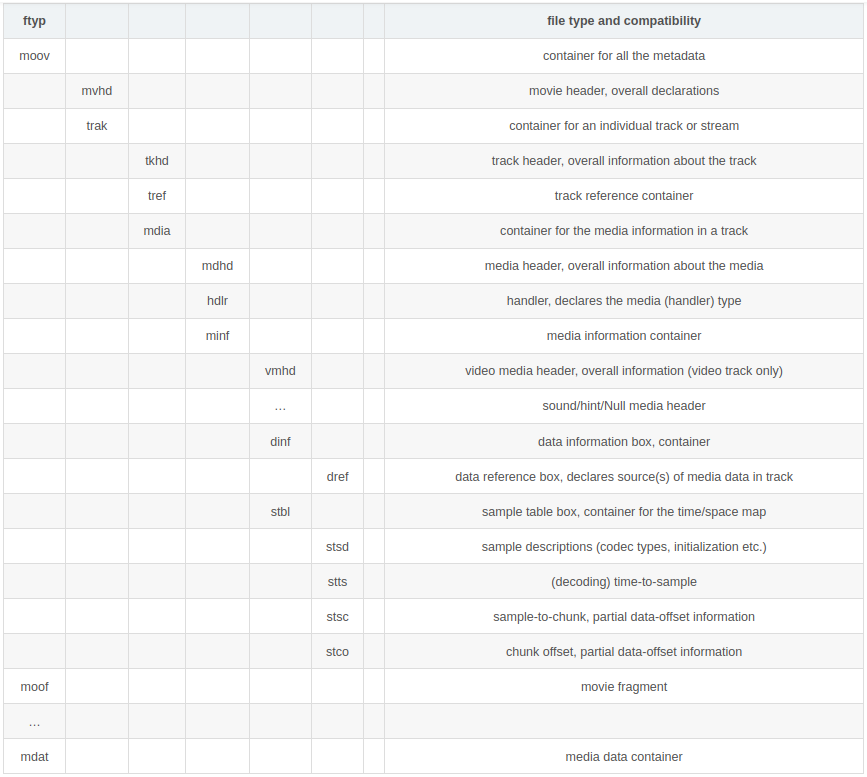
该 box只能被包含在文件层，被放在文件的最开始，而且不能被其他 box 包含。

1. 之后会有且只有一个“moov”类型的box（Movie Box），它是一种container box，子box包含了媒体的metadata信息；
2. MP4文件的媒体数据（最终解码播放的音视频数据）包含在“mdat”类型的box（Midia Data Box）中，该类型的box也是container box，一般有多个，也可以没有（当媒体数据全部引用其他文件时），媒体数据的结构由metadata进行描述。

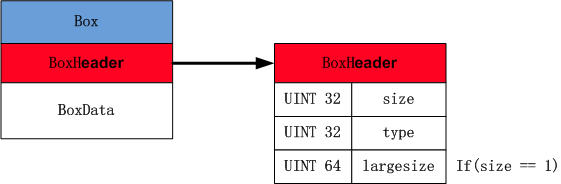
下面是一些概念：

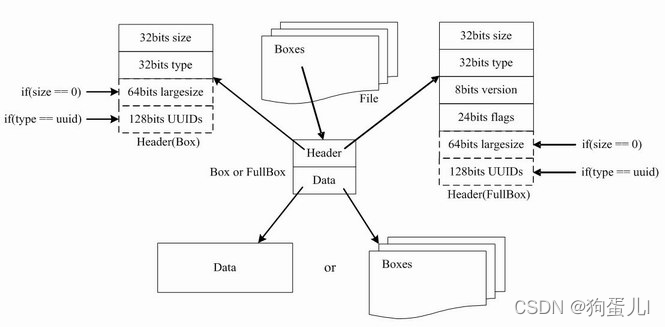
1. track 表示一些sample的集合，对于媒体数据来说，track表示一个视频或音频序列。
2. hint track 这个特殊的track并不包含媒体数据，而是包含了一些将其他数据track打包成流媒体的指示信息。
3. sample 对于非hint track来说，video sample即为一帧视频，或一组连续视频帧，audio sample即为一段连续的压缩音频，它们统称sample。对于hint tracksample定义一个或多个流媒体包的格式。
4. sample table 指明sampe时序和物理布局的表。
5. chunk 一个track的几个sample组成的单元。

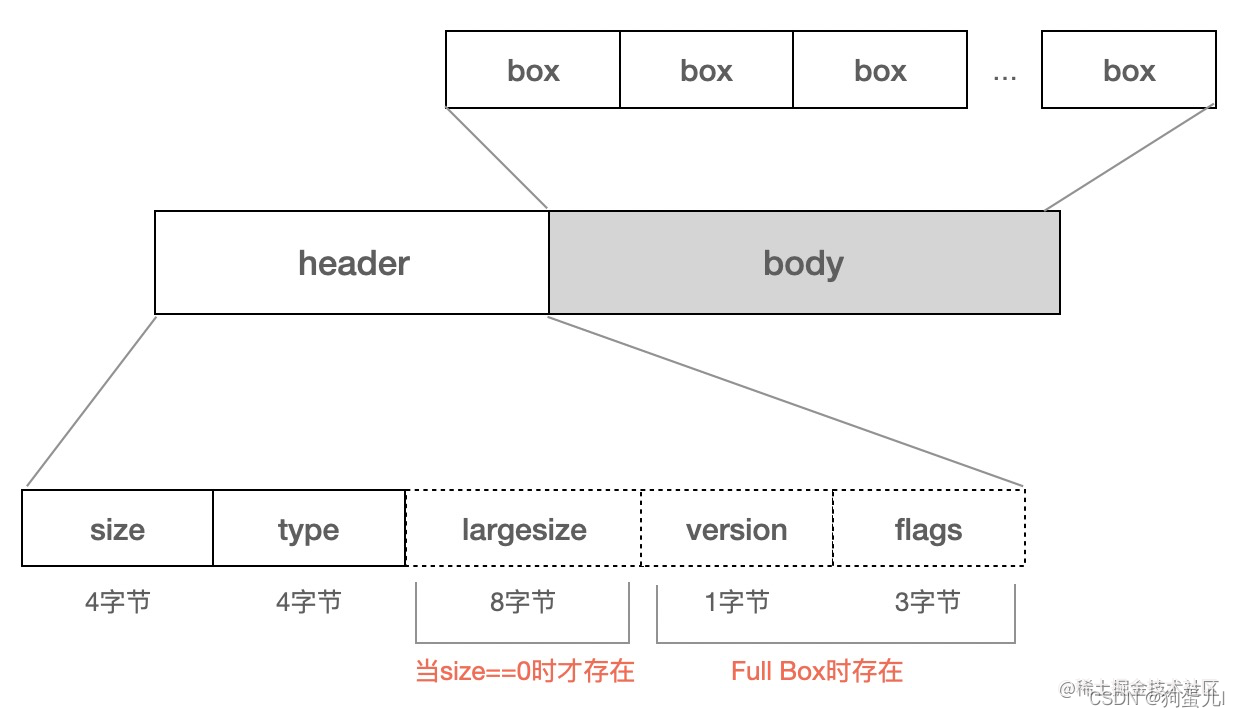
#### 重点关注的box



## 三、基本概念

1. 文件，由许多Box和FullBox组成。
2. Box，每个Box由Header和Data组成。
   * size：该大小包括 box header 和 box body 整个 box 的大小，这样我 们就可以在文件中定位各个 box。
   * type：size 后面紧跟的 32 位为 box type，一般是 4 个字符，如“ftyp”、“moov”等，这 些 box type 都是已经预定义好的，分别表示固定的意义。
   * largesize：如果box很大超过了uint32的最大数值，size就被设置为1，并用接下来的 largesize来存放大小。
3. FullBox，是Box的扩展，Box结构的基础上在Header中增加8bits version和24bits flags。
4. Header，包含了整个Box的长度size和类型type。
   * 当 size == 0时，代表这是文件中最后一个Box；
   * 当size==1时，意 味着Box长度需要更多bits来描述，在后面会定义一个64bits的largesize描述Box的长度；
   * 当type是uuid时，代表Box中的数据是用户自定义扩展类型。
5. Data，是Box的实际数据，可以是纯数据也可以是更多的子Boxes。
6. 当一个Box的Data中是一系列子Box时，这个Box又可成为Container Box。





## 四、重要box详解

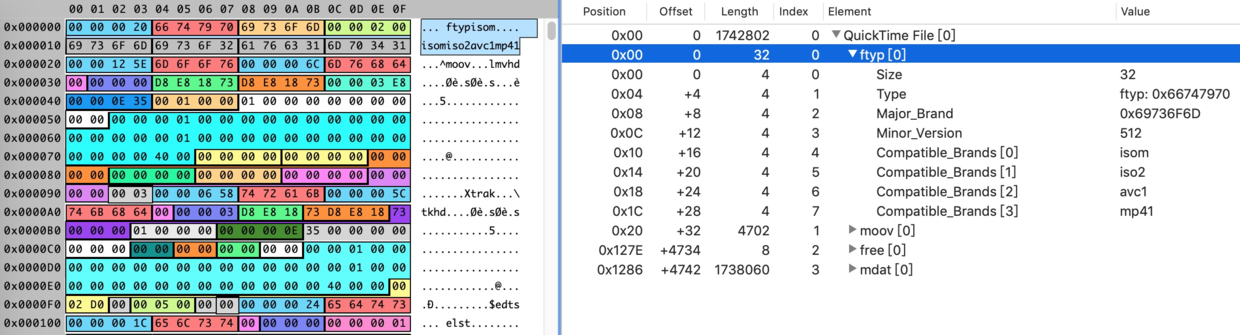
## File Type Box(ftyp)

ftyp body：依次包括 1 个 32 位的 major brand(4 个字符)，1 个 32 位的 minor version(整数)和 1 个以 32 位(4 个字符)为单位元素的数组 compatible brands。

isom（ISO Base Media file）是在 MPEG-4 Part 12 中定义的一种基础文件格式，MP4、3gp、QT 等常见的封装格式，都是基于这种基础文件格式衍生的。

MP4 文件可能遵循的规范有mp41、mp42，而mp41、mp42又是基于isom衍生出来的。

3gp(3GPP)：一种容器格式，主要用于3G手机上；

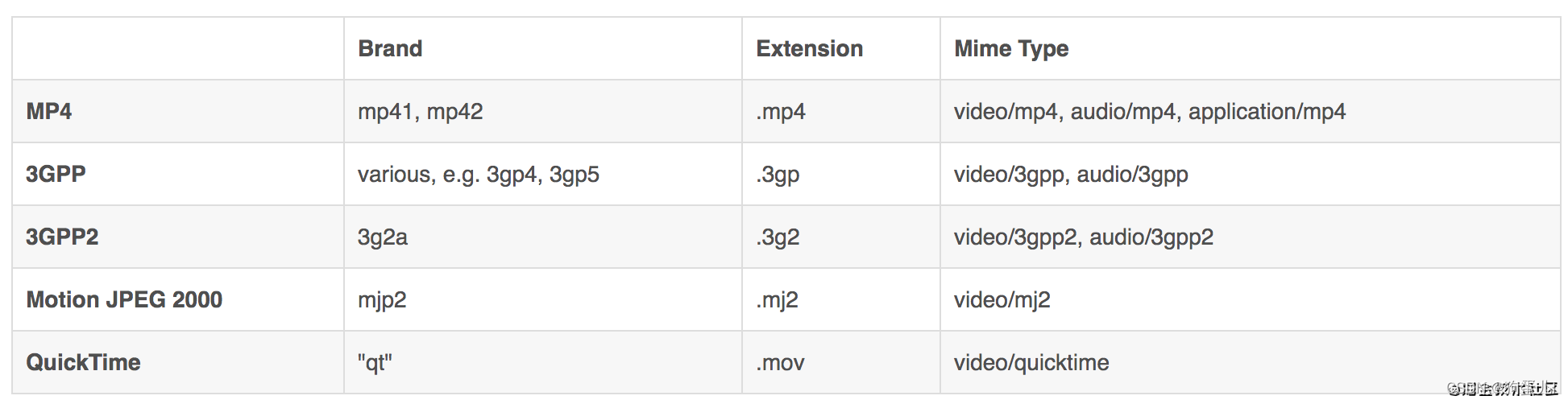
QT：QuickTime的缩写，.qt 文件代表苹果QuickTime媒体文件；

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； 在实际使用中，不能把 isom 做为major\_brand，而是需要使用具体的brand（比如mp41），因此，对于 isom，没有定义具体的文件扩展名、mime type。

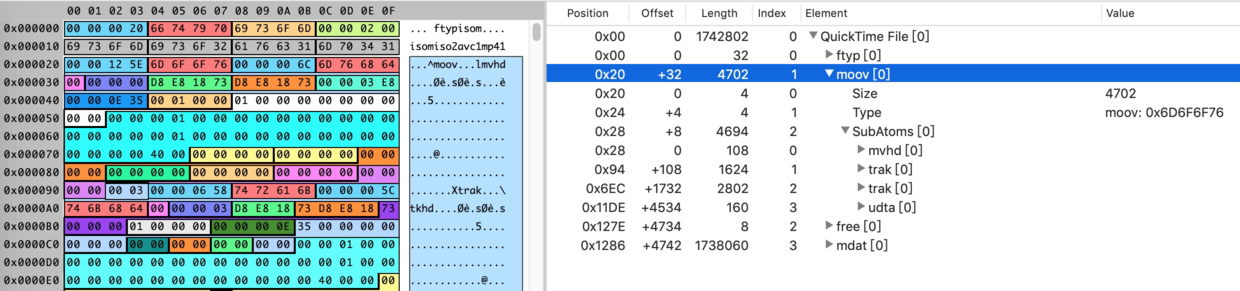
在实际使用中，不能把 isom 做为 major\_brand，而是需要使用具体的brand（比如mp41），因此，对于isom，没有定义具体的文件扩展名、mime type。



#### Movie Box(moov)

包含本文件中所有媒体数据的宏观描述信息以及每路媒体轨道的具体信息。一般位于 ftyp 之后，也有的视频放在文件末尾。注意，当改变 moov 位置时，内部一些值需要重新计算。

一般情况下，moov中会包含 1 个mvhd和若干个trak。其中mvhd为 header box，一般作为moov的第一个子 box 出现。



##### Movie Header Box(mvhd)

mvhd 定义了整个 movie 的特性，例如 time scale 和 duration。

creation\_time：文件创建时间；

modification\_time：文件修改时间；

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；

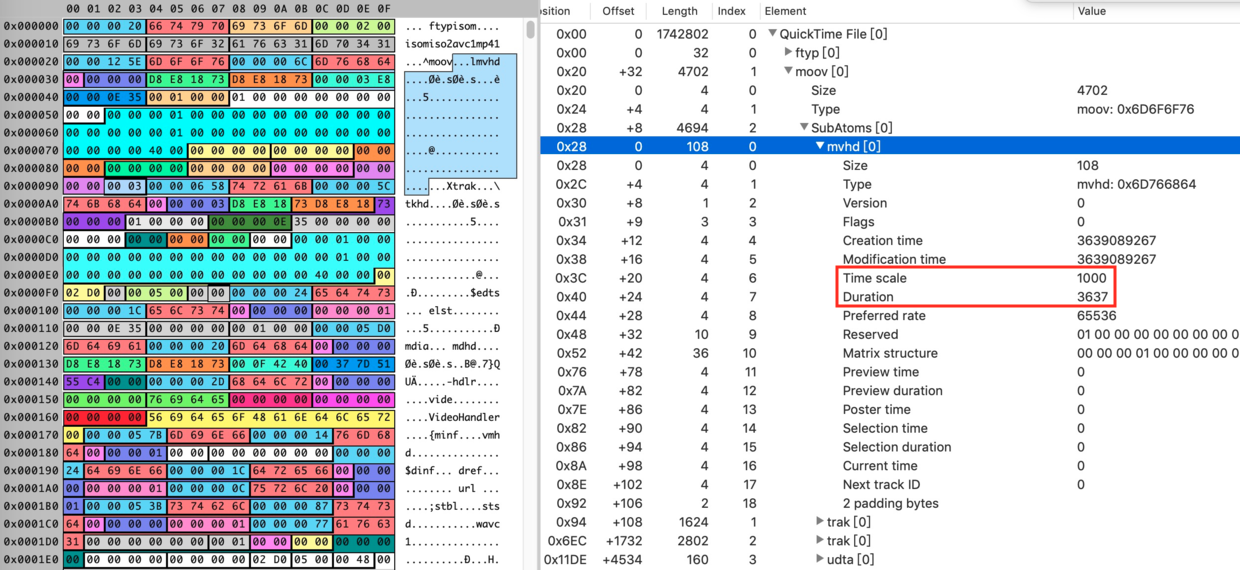
rate：推荐的播放速率，32位整数，高16位、低16位分别代表整数部分、小数部分（[16.16]），举例 0x0001 0000代表1.0，正常播放速度；

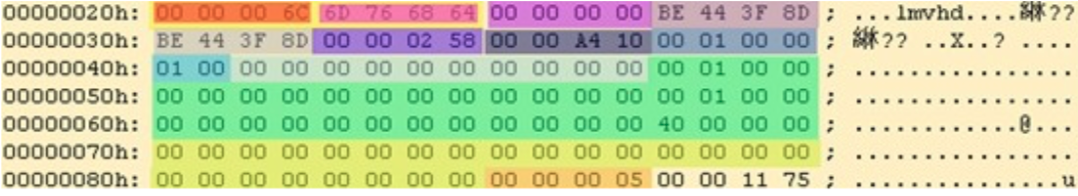
volume：播放音量，16位整数，高8位、低8位分别代表整数部分、小数部分（[8.8]），举例 0x01 00 表示1.0，即最大音量；

matrix：视频的转换矩阵，一般可以忽略不计；

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；

文件信息如时长为 3.637 秒：





##### Track Box(trak)

一个mp4文件可以包含一个或多个 tracks，它们之间相互独立，各自有各自的时间和空间信息。每个track box都有与之关联的media box。trak box 要求必须有一个 trak header box (tkhd) 和一个 media box(mdia)。

* Track Header Box(tkhd)

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；

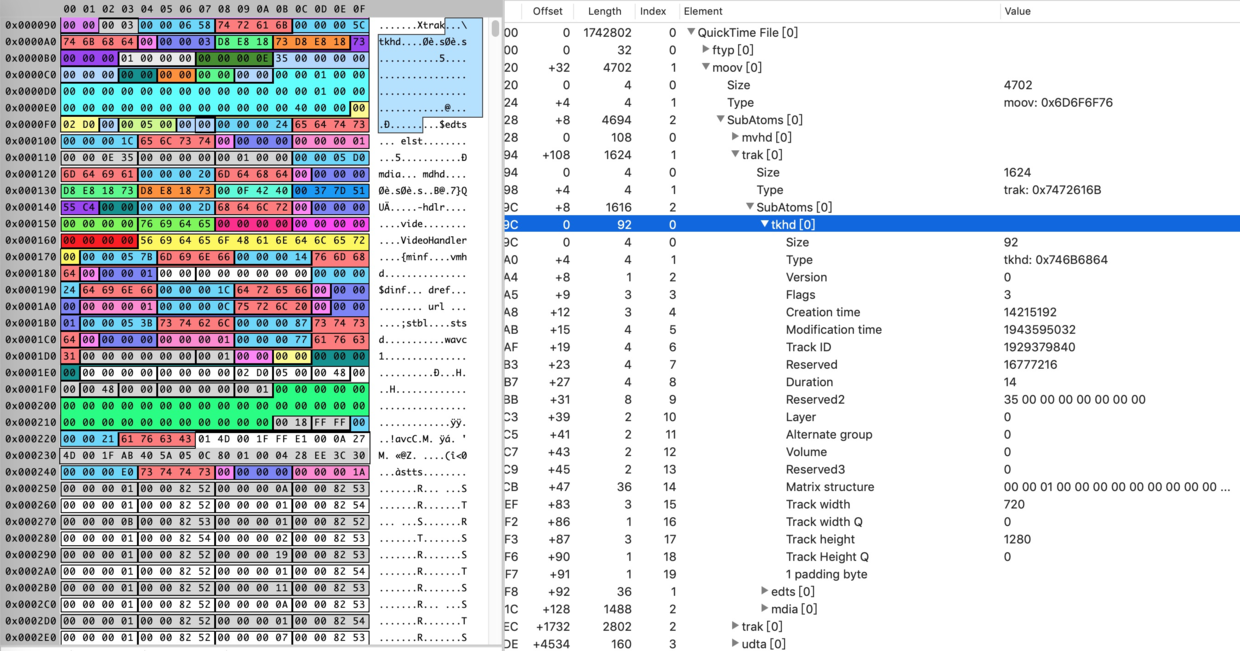
alternate\_group：当前track的分组ID，alternate\_group值相同的track在同一个分组里面。同个分组里的track，同一时间只能有一个track处于播放状态。当alternate\_group为0时，表示当前track没有跟其他track处于同个分组。一个分组里面，也可以只有一个track；

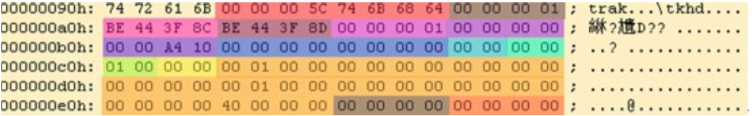
volume：audio track的音量，介于0.0~1.0之间；

matrix：视频的变换矩阵；

width、height：视频的宽高；

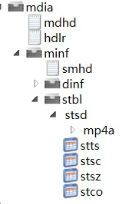






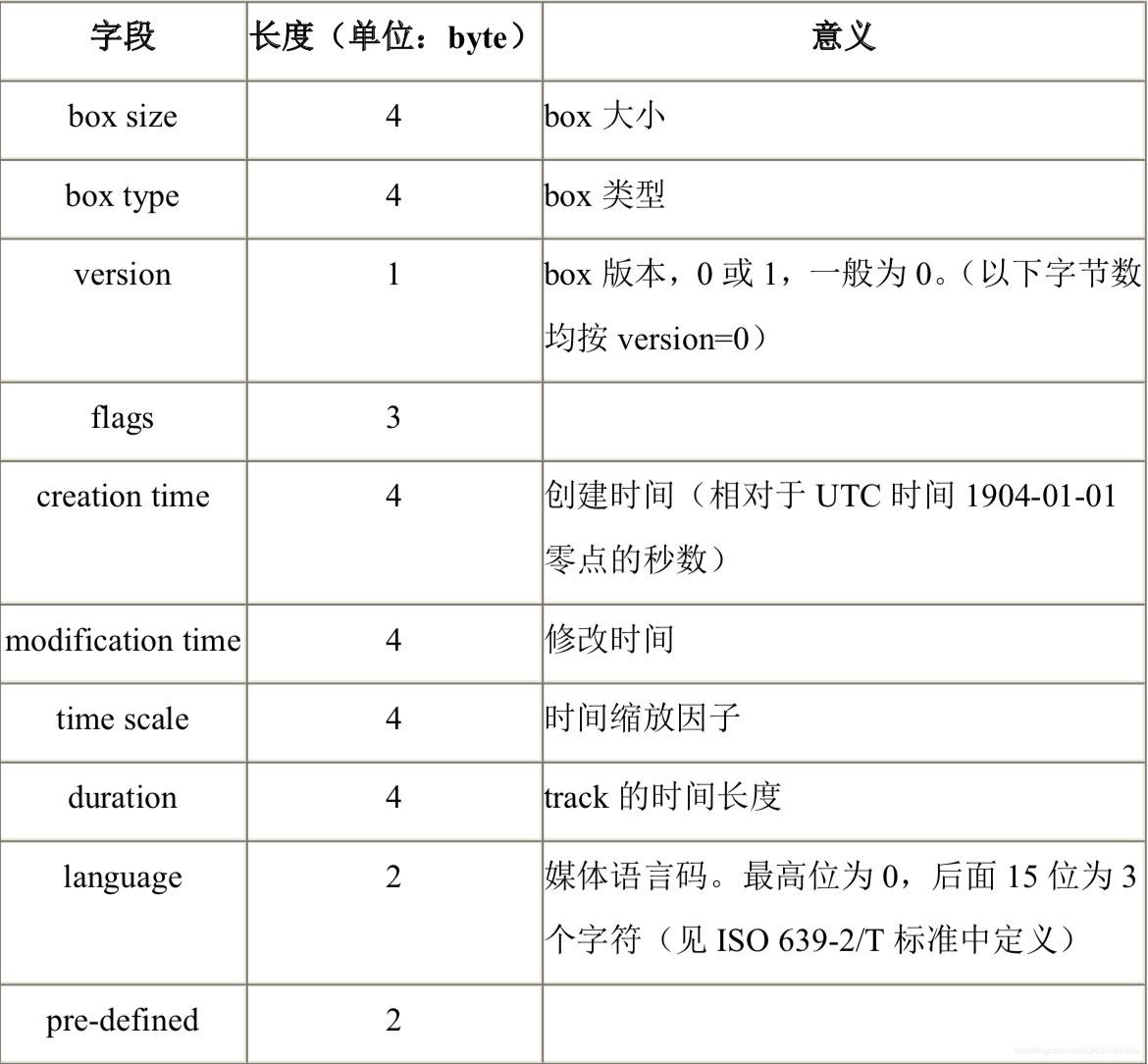
* Media Box(mdia)

mdia也是个 container box，其子 box 的结构和种类还是比较复杂的。总体来说，mdia定义了 trak 媒体类型以及 sample 数据，描述 sample 信息。一般mdia包含一个mdhd， 一个hdlr和一个minf，其中mdhd为 media header box，hdlr为 handler reference box，minf为 media information box。树形结构如下：



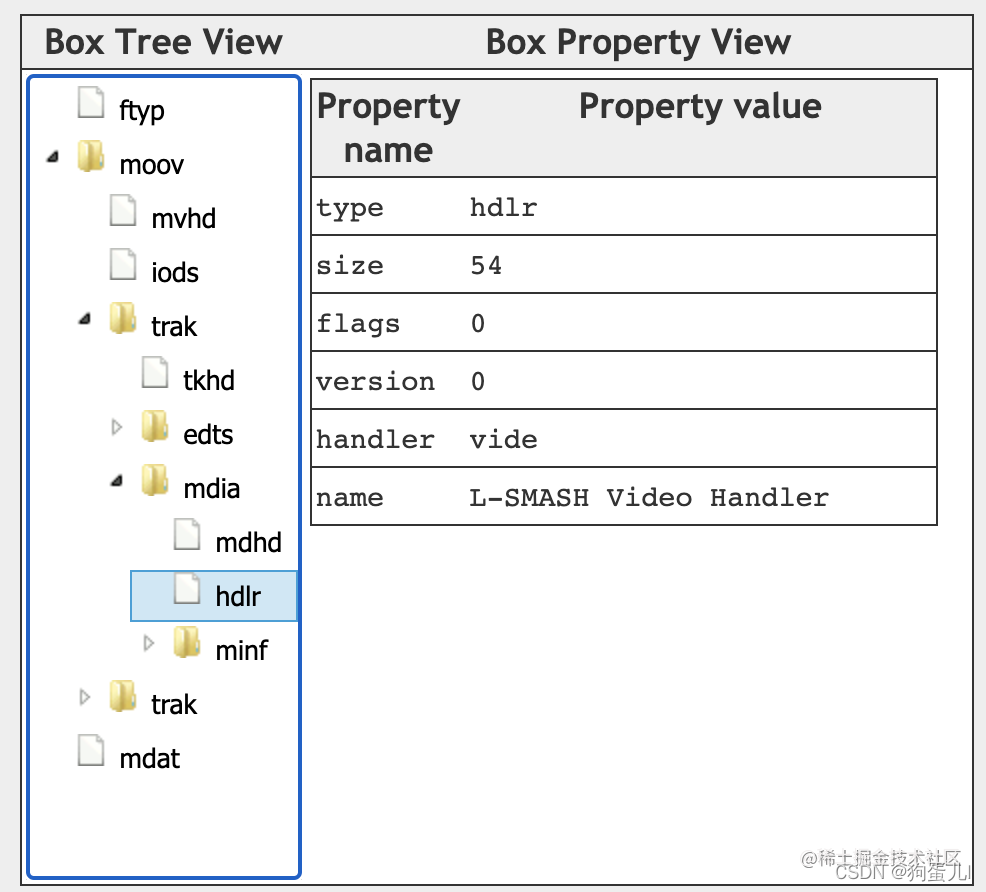
* + - Media Header Box(mdhd)

Media header box 定义了媒体的特性，例如 time scale 和 duration。



* + - Handler Reference Box(hdlr)

handler box 解释了媒体流的播放过程。例如，一个视频 handler 处理一个 video track。



* + - Media Information Box(minf)

minf存储了解释 trak 媒体数据的 handler-specific 信息，media handler用这些信息将媒体时间映射到媒体数据并进行处理。minf中的信息格式和内容与媒体类型以及解释媒体数据的 media handler密切相关，其他 media handler 不知道如何解释这些信息。minf是一个 container box，其实际内容由子 box说明。

一般情况下，minf包含一个 header box，一个dinf和一个stbl。

header box 根据track type(即 media handler type)分为vmhd、smhd，dinf为 data information box，stbl为 sample table box。

* + - * Video Media Header Box(vmhd)



* + - * Sound Media Header Box(smhd)



* + - * Data Information Box(dinf)

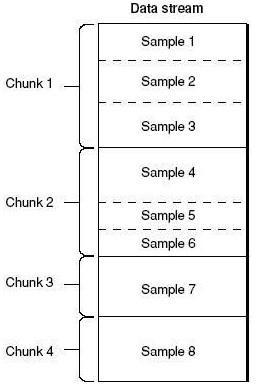
dinf解释如何定位媒体信息，是一个 container box。dinf一般包含一个dref，即 data reference box;dref下会包含若干个url或urn，这些 box 组成一个表，用来定位 trak 数据。简单的说，trak可以被分成若干段， 每一段都可以根据url或urn指向的地址来获取数据，sample 描述中会用这些片段的序号将这些片段组成一个完整的 trak。一般情况下，当数据被完全包含在文件中时，url或urn中的定位字符串是空的。



* + - * Sample Table Box(stbl)

MP4的媒体信息和数据是分开存放的。就是你想获得数据之前必须要解析出每个帧数据所在的位置。mp4存放这个帧信息的是放在stbl这个box里。而真实的数据放在mdat中。接下来就讲讲stbl与mdat的对应关系。

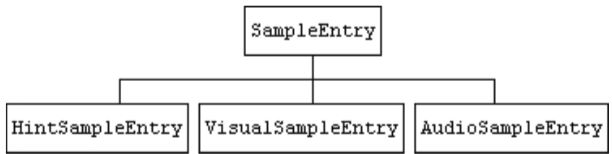
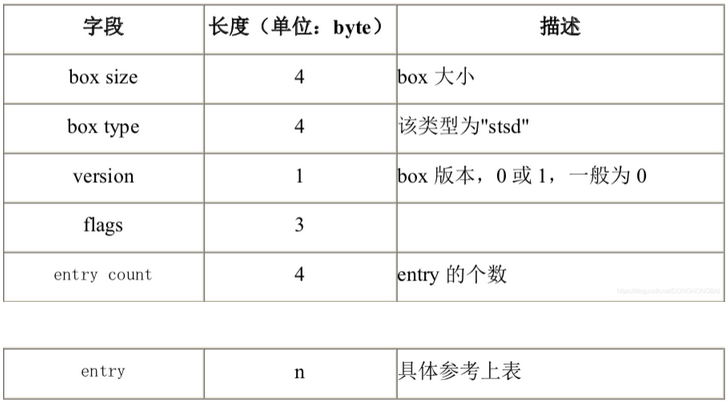
在 MP4 文件中，Sample 是一个媒体流的基本单元，例如视频流的一个 Sample 代表实际的 nal 数据。Chunk 是数据存储的基本单位，它是一系列 Sample 数据的集合，一个 Chunk 中可以包含一个或多的 Sample。



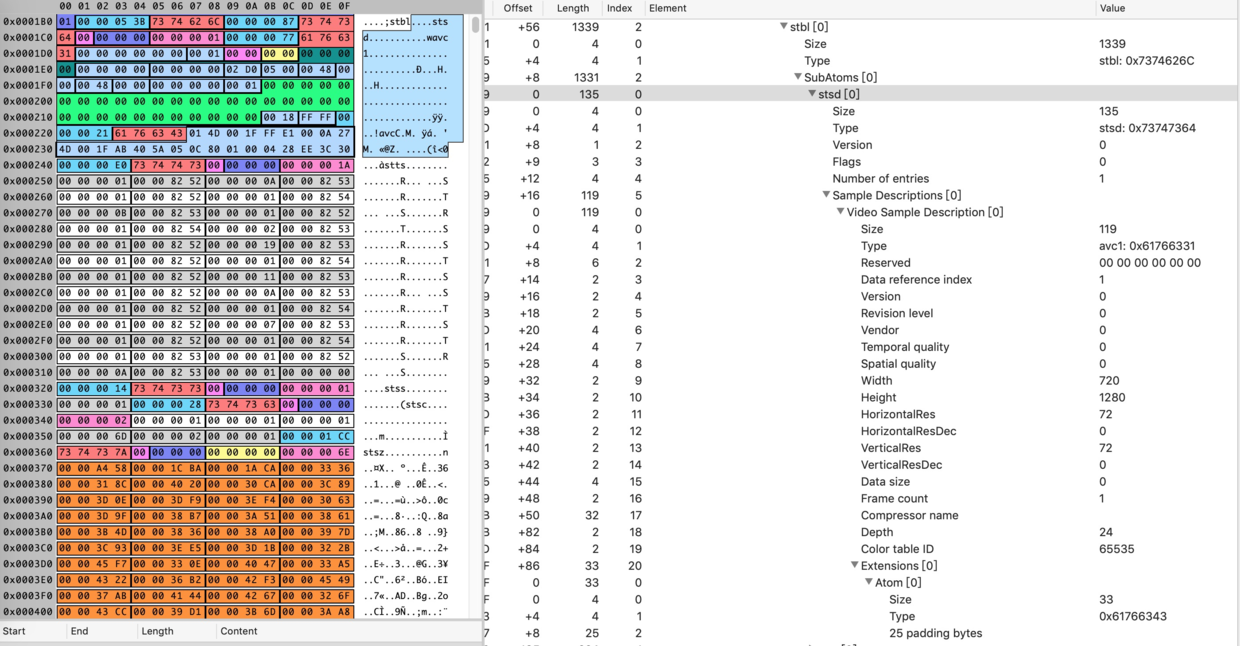
stbl包含了关于 trak 中 sample 所有时间和位置的信息，以及 sample 的编解码等信息。利用这个表，可以解释sample 的时序、类型、大小以及在各自存储容器中的位置。stbl是一个 container box。

* + - * + sample description box(stsd)

stsd必不可少，且至少包含一个条目，该 box 包含了 data reference box 进行 sample数据检索的信息。  
解析stsd可获得coding类型、视频宽高、音频samplesize、channelcount这些和解码器有关信息。

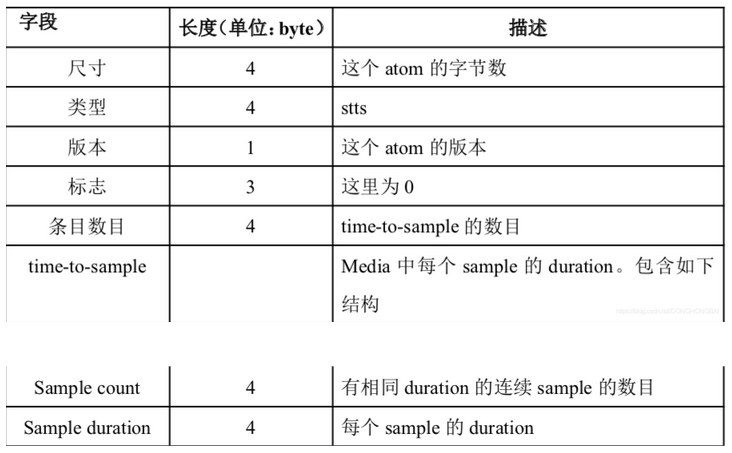


对于 h264 的视频流，其具体类型为 avc1，extensions 中存放有 sps，pps 等解码必要信息：



* + - * + time to sample box(stts)

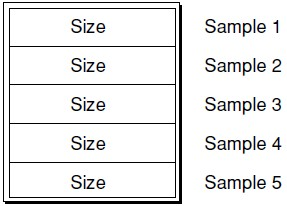
stts存储了 media sample的duration信息，提供了时间对具体data sample的映射方法，通过 这个atom，你可以找到任何时间的 sample。



* + - * + sample size box(stsz 或stz2)

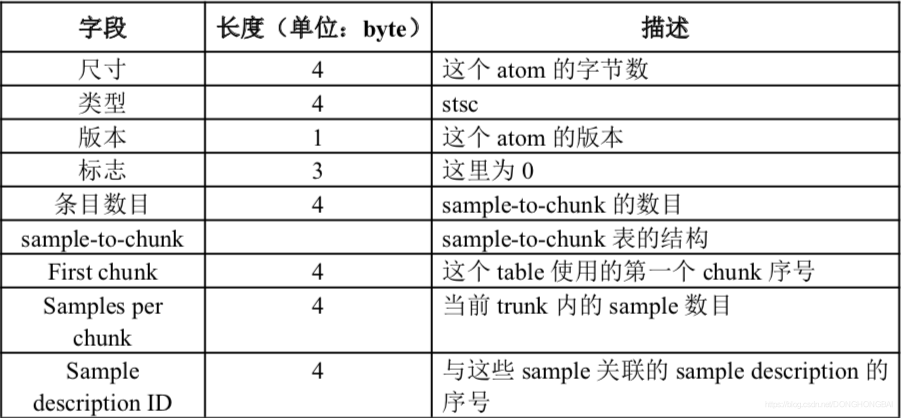
指定了每个 Sample 的 size。Sample Size Atom 包含两 Sample 总数和一张包含了每个 Sample Size 的表。

Sample Size 表的 entry 布局如下图：



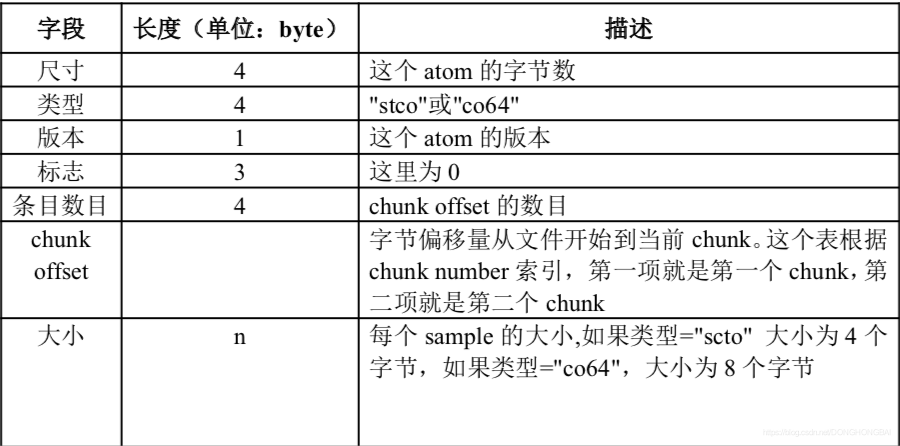
* + - * + sample to chunk box(stsc)

当添加 samples 到 media 时，用 chunks 组织这些 sample，这样可以方便优化数据获取。一个 trunk 包含一个或多个 sample，chunk 的长度可以不同，chunk 内的 sample 的长度也可以不同。sample-to-chunk box存储sample与chunk的映射关系。



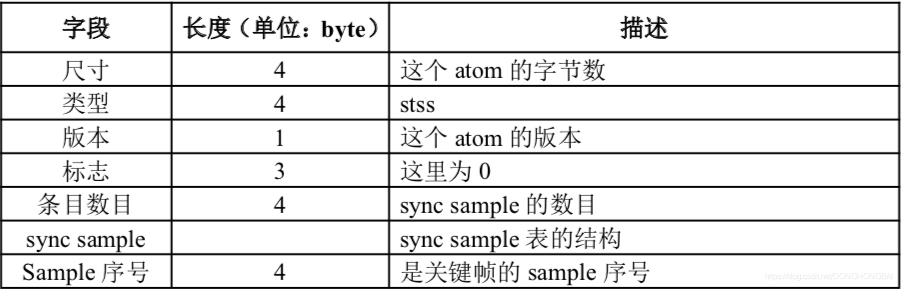
* + - * + chunk offset box (stco 或co64)

指定了每个 Chunk 在文件中的位置，这个表是确定每个 Sample 在文件中位置的关键。该表包含了 Chunk 个数和一个包含每个 Chunk 在文件中偏移位置的表。位置有两种可能，32位的和64位的，后者对非常大的电影很有用。



* + - * + composition time to sample box(ctts)
        + sync sample box(stss)

stss确定media中的关键帧。对于压缩的媒体，关键帧是一系列压缩序列的开始帧，它的解压缩是不依赖于以前的帧。后续帧的解压缩依赖于这个关键帧。



## **六、如何获取某一个 pts 对应的 Sample 在文件中的位置**

大体需要以下步骤：

1. 将 pts 转换到媒体对应的时间坐标系。
2. 根据 stts 计算某个 pts 对应的 Sample 序号。
3. 根据 stsc 计算 Sample 序号存放在哪个 Chunk 中。
4. 根据 stco 获取对应 Chunk 在文件中的偏移位置。
5. 根据 stsz 获取 Sample 在 Chunk 内的偏移位置并加上第 4 步获取的偏移，计算出 Sample 在文件中的偏移。

例如，想要获取 3.64 秒视频 Sample 数据在文件中的位置：

根据 time scale 参数，将 3.64 秒转换为视频时间轴对应的 3640000。

遍历累加下表所示 stts 所有项目，计算得到 3640000 位于第 110 个 Sample。

type stts

size 224

flags 0

version 0

sample\_counts 1,10,1,1,11,1,1,2,1,25,1,1,1,17,1,10,1,1,1,7,1,1,1,1,10,1

sample\_deltas 33362,33363,33362,33364,33363,33362,33364,33363,33362,33363,33362,33364,33362,33363,33362,33363,33362,33364,33362,33363,33362,33364,33363,33362,33363,0

查询下表所示 stsc 所有项目，计算得到第 110 个 Sample 位于第 109 个 Chunk，并且在该 Chunk 中位于第 2 个 Sample。

type stsc

size 40

flags 0

version 0

first\_chunk 1,109

samples\_per\_chunk 1,2

sample\_description\_index 1,1

查询下表所示 stco 所有项目，得到第 109 个 Chunk 在文件中偏移位置为 1710064。

Property name Property value

type stco

size 452

flags 0

version 0

chunk\_offsets

4750,47007,54865,61967,75519,88424,105222,117892,133730,149529,165568,182034,194595,210776,225470,240756,255358,270711,285459,300135,315217,330899,347372,363196,376409,394509,407767,424615,438037,455603,469784,487287,505197,519638,536714,553893,567187,584744,599907,615298,630669,645918,662605,678655,693510,708980,724061,738946,754170,771520,787233,800847,816997,832490,847814,862559,877929,898379,911054,925810,943883,956497,974403,991527,1009478,1025198,1041806,1062609,1078401,1091360,1105142,1118748,1132815,1145281,1156966,1171871,1186742,1202760,1218235,1236688,1249330,1263163,1280880,1297903,1313162,1332885,1345726,1359017,1376283,1391401,1405512,1419550,1433644,1452103,1475241,1492689,1511291,1522606,1535368,1559413,1575331,1588853,1609829,1626623,1642798,1658640,1674160,1693972,1710064

查询下表所示 stsz 所有项目，得到第 109 个 Sample 的 size 为 14808。计算得到 3.64 秒视频 Sample 数据在文件中：

offset：1710064 + 14808 = 1724872 size：17930

type stsz

size 460

flags 0

version 0

sample\_sizes

42072,7354,6858,13110,12684,16416,12490,15497,15630,15865,16116,12387,15775,14519,14929,14433,15181,14390,14496,14717,15507,16101,15643,12843,17911,13070,16455,13221,17186,14002,17139,17737,14251,16708,16999,12911,17356,14801,15213,15016,15062,16505,15689,14657,15053,14907,14527,15048,17161,15308,13432,15777,15307,14971,14568,14987,20264,12494,14382,17873,12235,17718,16770,17766,15366,16420,20623,15403,12761,13394,13390,13714,12295,11505,14541,14689,15635,15291,18091,12458,13645,17346,16847,14902,19530,12446,13105,16872,14937,13944,13657,13908,18092,22959,17080,18421,11129,12400,23844,15564,13340,20603,16609,15984,15474,15339,19451,15719,14808,17930

sample\_size 0

sample\_count 110

验证：用编辑器打开 MP4 文件，定位到文件偏移 offset = 1724872 的位置，前 4 字节值为 0x00004606。在 avcc 中一个 Sample 的前 4 个字节代表这个包的大小，转换为十进制是 17926，该值正好等于 size = 17930 减去表示长度的四个字节。

## 七、fMP4（Fragmented mp4）

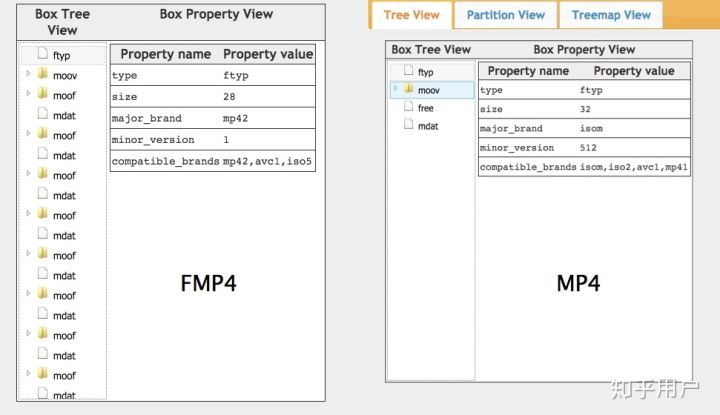
fMP4 跟普通 mp4 基本文件结构一样。普通mp4用于点播场景，fmp4通常用于直播场景。

它们有以下差别：

* 普通mp4的时长、内容通常是固定的。fMP4 时长、内容通常不固定，可以边生成边播放；
* 普通mp4完整的metadata都在moov里，需要加载完moov box后，才能对mdat中的媒体数据进行解码渲染；

fmp4不需要moov Box来initalization，fmp4的moov Box只包含了一些track信息。

* fMP4中，媒体数据的metadata在moof box中，moof 跟 mdat （通常）结对出现。moof 中包含了sample duration、sample size等信息，因此，fMP4可以边生成边播放；是一个流式的封装格式。

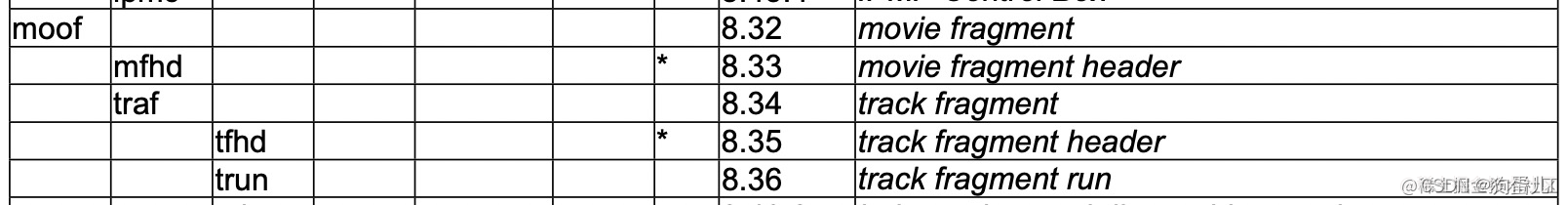


怎么判断mp4文件是普通mp4，还是fMP4呢？

一般可以看下是否存在存在mvex（Movie Extends Box）。

当存在mvex时，表示当前文件是fmp4（非严谨）。此时，sample相关的metadata不在moov里，需要通过解析moof box来获得。

# moof（Movie Fragment Box）

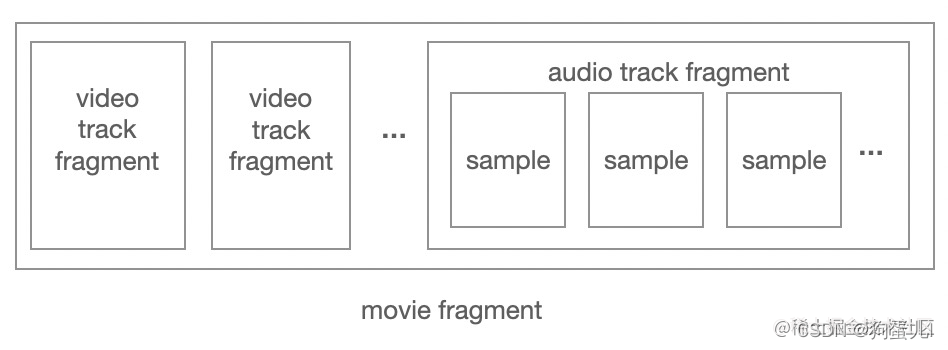
moof是个container box，相关 metadata 在内嵌box里，比如 mfhd、 tfhd、trun 等。

mfhd（Movie Fragment Header Box）

结构比较简单，sequence\_number 为 movie fragment 的序列号。根据 movie fragment 产生的顺序，从1开始递增。

traf（Track Fragment Box）

对 fmp4 来说，数据被氛围多个 movie fragment。一个 movie fragment 可包含多个track fragment（每个 track 包含0或多个 track fragment）。每个 track fragment 中，可以包含多个该 track 的 sample。

每个 track fragment 中，包含多个 track run，每个 track run 代表一组连续的sample。

tfhd（Track Fragment Header Box）

tfhd 用来设置 track fragment 中 的 sample 的 metadata 的默认值。