**MP4结构**

1. 简述

MP4(MPEG-4 Part 14)是在ISO/IEC 14496-14标准文件中定义的，属于MPEG-4的一部分。MP4是一种描述较为全面的容器格式，被认为可以在其中嵌入任何形式的数据，各种编码的视频、音频等，常见的大部分的MP4文件存放的AVC(H.264)或MPEG-4(Part 2)编码的视频和AAC编码的音频。MP4格式的官方文件后缀名是.mp4，还有其他的以mp4为基础进行的扩展或者是阉割版的格式，如：M4V、3GP、F4V等。

mp4是由一个个box组成的，大box中存放小box，一级嵌套一级来存放媒体信息。box的基本结构ftyp box、moov box、mdat box等。一般要把moov放在mdat前面达到快速起播。

一个 box 是由 box header 和 box 里面包含的数据组成。

box header：包含了 box 的大小size和类型type等信息。其中，size指明了整个box所占用的大小（包括 header 部分）：  
（1）size = 0 表示该 box 是最后一个 box；  
（2）如果 box 很大（例如存放具体视频数据的 mdat box），超过了 uint32 的最大数值，size 就被设置为 1，并用接下来的 uint64 的 largesize 来存放大小。

type = uuid 表示该 box 中的数据是用户自定义扩展类型。box 中的字节序为网络字节序，也就是大端字节序(Big-Endian)。box 根据 header 部分包含的信息的不同可以分为 Box 和 Full Box。Full box多了version和flags。

1. ftyp

File type box是MP4文件的第一个box，在根目录。Type是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。

1. free

Free Space box中的内容是无关紧要的，可以被忽略。该box被删除后，不会对播放产生任何影响，它的type域可以是free或skip。

四、mdat

Media data box是文件层 box，可以有多个，也可以没有（当媒体数据全部为外部文件引用时），用于存储媒体数据，type是mdat。

数据直接跟在box type字段后面，它的结构是由metadata来描述的，metadata通过文件中的绝对偏移来引用媒体数据。

五、moov

Movie box是文件层box，唯一。存放metadata信息，检索mdat中chunk和sample数据。

一般情况下，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。trak包含了一个track的相关信息，是一个container box。

1、mvhd

Size：box大小。

Type：mvhd。

Version：版本。

Flags：扩展标志位。

creation\_time;创建时间，相对于UTC时间1904-01-01零点的秒数。

modification\_time;修订时间。

timescale;时间基准，在1秒时间内的刻度值，可以理解为1秒长度的时间单元数，一般情况下视频的都是90000。

duration;文件时长，以timescale为基准。

Rate播放速率，为1.0表示正常播放。

Volume 1.0表示最大音量。

Reserved

matrix视频变换矩阵

pre\_defined

next\_track\_ID下一个track使用的id号

2、trak

track box是一个container box，其子box包含了该track的媒体数据引用和描述。一个MP4文件中的媒体有一个或多个track，这些track之间彼此独立，有自己的时间和空间信息。必须包含一个tkhd box和一个mdia box。Type是trak。

2.1 tkhd

Size：box大小。

Type：tkhd。

Version：版本。

Flags：播放时是否被引用。

track\_ID：id号，不能重复且不能为0。

reserved

Duration track的时长，以timescale为基准

reserved

layer 视频层，默认为0

alternate\_group

volume

reserved

matrix

Width 宽

Height 高

2.2、mdia（Media）

包含了该 track 的媒体信息，比如媒体类型和 sample 信息。其必须包含：  
（1）一个媒体头 mdhd；  
（2）一个句柄参考 hdlr；  
（3）一个媒体信息 minf 和 udta。

2.2.1、mdhd（Media Header）

包含了该 track 的总体信息，mdhd 和 tkhd 内容大致都是一样的。tkhd 通常是对指定的 track 设定相关属性和内容，而 mdhd 是针对于独立的 media 来设置的，一般情况下二者相同。

2.2.2、minf（Media Information）

minf box 包含了所有描述该 track 中的媒体信息的对象，信息存储在其子 box 中。

2.2.2.1、vmhd（Video Media Information Header）

用在视频 track 中，包含当前 track 的视频描述信息，如视频编码等信息。

2.2.2.2、smhd（Sound Media Information Header）

用在音频 track 中，包含当前 track 的音频描述信息，如编码格式等信息。

2.2.2.3、stbl

stbl包含了关于track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，其子box包括：

stts：Decoding Time to Sample Box 时间戳和Sample映射表  
stsd：Sample Description Box  
stsz, stz2：Sample Size Boxes 每个Sample大小的表。  
stsc：Sample to chunk 的映射表。  
stco，co64：Chunk位置偏移表  
stss：关键帧index。

A、Sample Description Box（stsd）

stsd必不可少，且至少包含一个条目。没有stsd就无法计算media sample的存储位置。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。

B、Time To Sample Box（stts）

stts存储了sample的duration，描述了sample时序的映射方法，我们通过它可以找到任何时间的sample。stts可以包含一个压缩的表来映射时间和sample序号，用其他的表来提供每个sample的长度和指针。表中每个条目提供了在同一个时间偏移量里面连续的sample序号，以及samples的偏移量。递增这些偏移量，就可以建立一个完整的time to sample表。

C、Sync Sample Box（stss）

stss确定media中的关键帧。对于压缩媒体数据，关键帧是一系列压缩序列的开始帧，其解压缩时不依赖以前的帧，而后续帧的解压缩将依赖于这个关键帧。stss可以非常紧凑的标记媒体内的随机存取点，它包含一个sample序号表，表内的每一项严格按照sample的序号排列，说明了媒体中的哪一个sample是关键帧。如果此表不存在，说明每一个sample都是一个关键帧，是一个随机存取点。

D、Sample To Chunk Box（stsc）

用chunk组织sample可以方便优化数据获取，一个chunk包含一个或多个sample。stsc中用一个表描述了sample与chunk的映射关系，查看这张表就可以找到包含指定sample的chunk，从而找到这个sample。

box header和version字段后会有一个entry count字段，根据entry的个数，每个entry会有type信息，如vide、sund等，根据type不同sample description会提供不同的信息，例如对于video track，会有VisualSampleEntry类型信息，对于audio track会有AudioSampleEntry类型信息。

每个entry 表示着一组数据，entry\_count 表示这数量。这一组其实是相同类型的chunk。first\_chunk 表示 这一组相同类型的chunk中 的第一个chunk数。这些chunk 中包含的Sample 数量，即samples\_per\_chunk 是一致的。每个Sample 可以通过sample\_description\_index 去stsd box 找到描述信息。

E、Sample Size Box（stsz）

stsz定义了每个sample的大小，包含了媒体中全部sample的数目和一张给出每个sample大小的表。这个box相对来说体积是比较大的。

F、Chunk Offset Box（stco）

stco定义了每个chunk在媒体流中的位置。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。在一个表中只会有一种可能，这个位置是在整个文件中的，而不是在任何box中的，这样做就可以直接在文件中找到媒体数据，而不用解释box。需要注意的是一旦前面的box有了任何改变，这张表都要重新建立，因为位置信息已经改变了。

表格解析：

1. stsd

第一步解析出entries个数，根据track类型确定stsd类型：video、audio、subtitle等。

根据stsd类型，按格式读取codec\_id和视频宽高、音频采样率声道数等参数。多个entries说明track中有不同编码参数的sample，一般每个track只有一种sample类型。

1. stts

第一步解析出entries个数。一般entries是1，因为每一帧的时长一般是相同的。

for (i = 0; i < entries && !pb->eof\_reached; i++) {

int sample\_duration;这一类sample的时长

unsigned int sample\_count; sample这一类sample的个数

}

根据这个表可以定位每个sample在track中的时间位置。

1. stsz

第一步解析出entries个数，就是sample的个数，记录每一个sample的大小，单位是字节，累加计算数据部分的大小。

for (i = 0; i < entries && !pb->eof\_reached; i++) {

sc->sample\_sizes[i] = get\_bits\_long(&gb, field\_size);

sc->data\_size += sc->sample\_sizes[i];

}

1. stsc

第一步解析出entries个数，chunk组的个数，每一组拥有同样个数的sample。

for (i = 0; i < entries && !pb->eof\_reached; i++) {

sc->stsc\_data[i].first = avio\_rb32(pb);

sc->stsc\_data[i].count = avio\_rb32(pb);

sc->stsc\_data[i].id = avio\_rb32(pb);

}

first表示这一组相同类型的chunk中的第一个chunk的下标索引值。这些chunk中包含的Sample 数量，即count是一致的。每个Sample 可以通过id去stsd box找到编码参数的描述信息。

5、stco

第一步解析出entries个数，chunk的个数，记录每一个chunk再文件的偏移位置。

for (i = 0; i < entries && !pb->eof\_reached; i++) {

sc->chunk\_offsets[i] = avio\_rb32(pb);

}

6、总结，从mp4文件提取媒体数据的步骤：

解析stco得到chunk表，知道chunk的总数及每个chunk所在文件的位置偏移量。

解析stsc 配合着上面的stco表，得到每个Sample与chunk的关系表。根据stsz（每个sample的大小），加上chunk的偏移位置获得每个Sample的偏移位置信息。并获取sample的编码参数。

配合上面的stts时间表，得到每个sample对应的时间位置，再根据编码参数等，找到某个时间位置的ES流。

7、stss关键帧index。

for (i = 0; i < entries && !pb->eof\_reached; i++) {

sc->keyframes[i] = avio\_rb32(pb);

}

entries是关键帧的个数，记录每一个关键帧的sample下标，关键帧和sample一一对应。Seek时先根据seek时间找到某一个sample，然后根据这个sample的下标去关键帧列表中找到最近的一个，确定要seek到哪个关键帧sample，再根据上面计算出来的sample与文件偏移位置的表，seek到文件的具体位置。