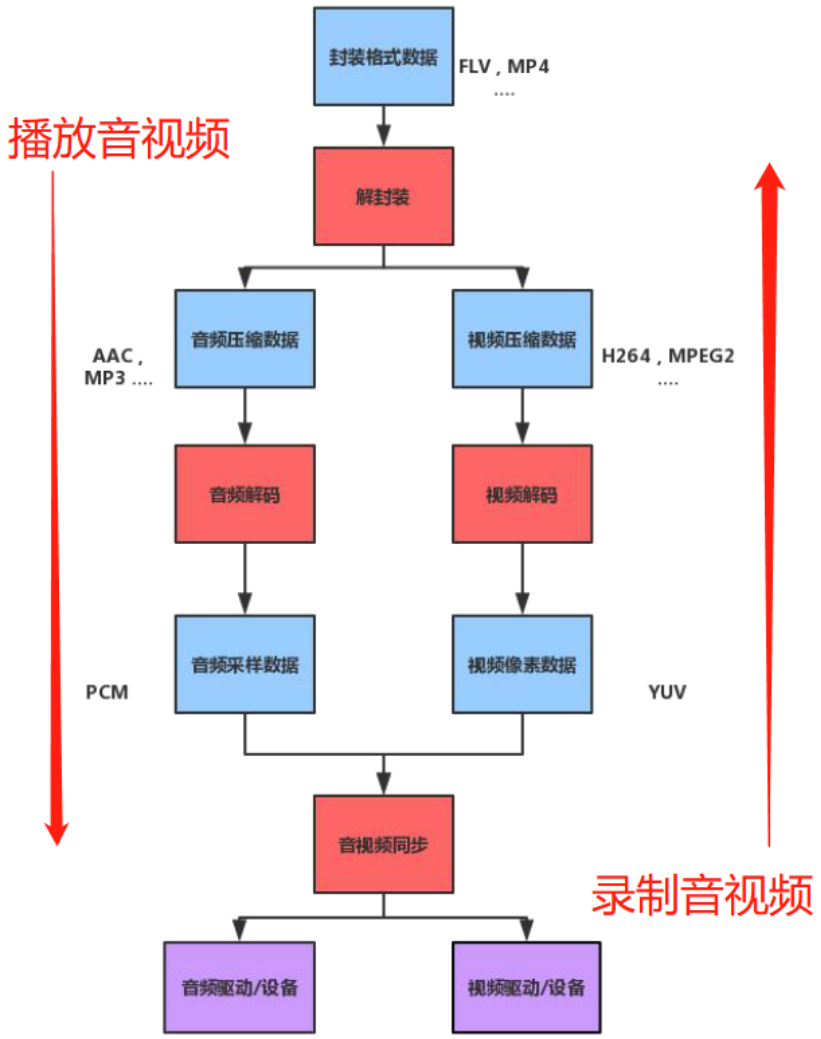
1 FFmpeg 概述

> 引自 **百度百科：FFmpeg**
>
> FFmpeg 是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用 LGPL 或 GPL 许可证。它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库 libavcodec，为了保证高可移植性和编解码质量， libavcodec 里很多 code 都是从头开发的。
>
> FFmpeg 在 Linux 平台下开发，但它同样也可以在其它操作系统环境中编译运行，包括 Windows、Mac OS X 等。这个项目最早由 Fabrice Bellard 发起，2004年至2015年间由 Michael Niedermayer 主要负责维护。许多 FFmpeg 的开发人员都来自 MPlayer 项目，而且当前 FFmpeg 也是放在 MPlayer 项目组的服务器上。项目的名称来自 MPEG **视频编码标准**，前面的 "FF" 代表 "Fast Forward" 。

2 音视频各阶段编解码格式

2.1 播放与录制音视频流程



2.1 视频

2.1.1 视频像素格式

- **ARGB_8888**：常见的图像像素编码格式，用 4 个字节分别存储图像的 A 透明度信息， R 红色通道信息， G 绿色通道信息， B 蓝色通道信息。
- **YUV**：YUV像素格式来源于RGB像素格式，通过公式运算，YUV三分量可以还原出RGB，YUV转RGB的公式如下：

```
R = Y + 1.403V
G = Y - 0.344U - 0.714V
B = Y + 1.770U
```

一般，将 RGB 和 YUV 的范围均限制在 [0, 255] 间，则有如下转换公式：

```
R = Y + 1.403(V - 128)
G = Y - 0.344(U - 128) - 0.714(V - 128)
B = Y + 1.770(U - 128)
```

YUV 格式根据采样方式不同又分为 YUV444 , YUV422 , YUV420 ， 详情参考 **这篇博客**。

2.1.2 视频压缩格式

压缩格式	适用静态图或视频?	特征描述	应用场景
JPEG / JPEG 2000	静态图	JPEG / JPEG 2000 利用的压缩算法不同	DSC 等
M-JPEG	视频	只是由连续的JPEG图像组成	PC-CAM ， 监控等
MPEG1	视频	MPEG 组织定义的 VCD 应用程序的标准	VCD 等
MPEG2	视频	MPEG 组织定义的 DVD ， DVB 应用程序的标准	DVD ， DVB ， 监控等
MPEG4	视频	MPEG 组织定义的网络应用程序的标准	监控，VOD ， IPTV ， PMP 等
H.261	视频	国际电联定义**基本**互联网多媒体应用标准	监控，视频会议等
H.263	视频	国际电联定义**低带宽**互联网多媒体应用标准	监控，视频会议等
H.264	视频	具有更高压缩效率的下一代标准	IPTV ， 监控 ， 多媒体等
WMV9	视频	微软定义的下一代标准	IPTV ， VOD
AVS1.0	视频	中国自有专利标准	

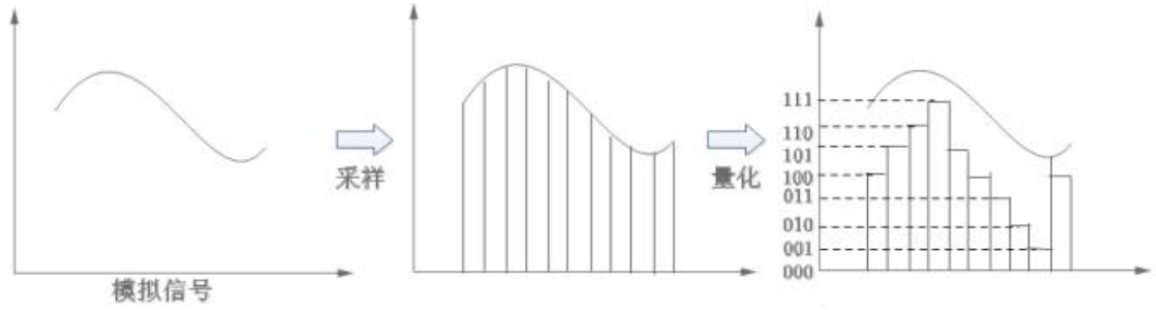
2.2 音频

2.2.1 音频采样数据格式

> 参考了 Allan_Wang 的 **音视频学习框架**。

- **采样**
  所谓采样就是在时间轴上对模拟信号进行数字化，根据采样定理（香农采样定理，奈奎斯特采样定理），按照比声音最高频率 2 倍以上的频率进行采样（AD转换）。人耳听力范围是 20Hz~20kHz，所以采样频率一般为 40kHz 左右，常用的有 44.1kHz (44100 次/s 采样)、 48kHz 等，**采样率越高，音质越好。**

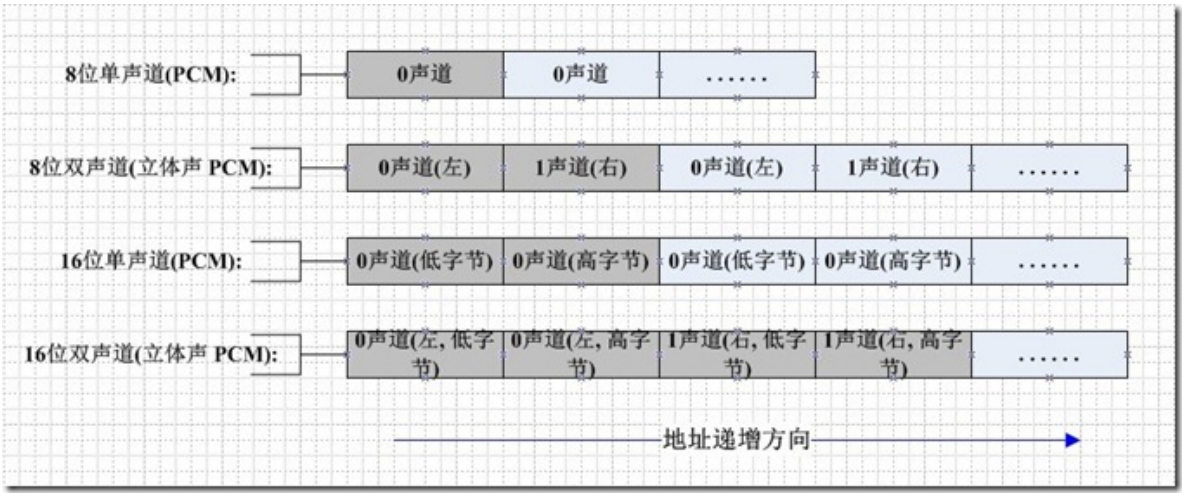
- **量化**
  是指在幅度轴上对信号进行数字化。对模拟音频信号的幅度进行数字化，它决定了模拟信号数字化以后的动态范围，常用的有8位、12位和16位。**量化位越高，信号的动态范围越大，数字化后的音频信号就越可能接近原始信号。**



- **PCM 编码**

PCM（Pulse Code Modulation）也被称为脉冲编码调制。PCM 音频数据是**未经压缩的音频采样数据裸流**，它是由模拟信号经过采样、量化、编码转换成的标准的数字音频数据。

如果是单声道的音频文件，采样数据按时间的先后顺序依次存入（有的时候也会采用 LRLRLR 方式存储，只是另一个声道的数据为 0），如果是双声道的话就按照 LRLRLR 的方式存储，存储的时候还和机器的大小端有关。大端模式如下图所示：



一般我们描述PCM音频数据的参数的时候有如下描述方式

> 44100HZ 16bit stereo : 每秒钟有 44100 次采样，采样数据用 16 位(2 字节)记录, 双声道(立体声);
> 22050HZ 8bit mono : 每秒钟有 22050 次采样，采样数据用 8 位(1 字节)记录, 单声道。

44100Hz 指的是采样率，它的意思是每秒取样44100次。采样率越大，存储数字音频所占的空间就越大。

16bit 指的是采样精度，意思是原始模拟信号被采样后，每一个采样点在计算机中用 16 位（两个字节）来表示。采样精度越高越能精细地表示模拟信号的差异。

2.2.2 音频压缩数据格式

格式	描述	备注
MP3	全名是 **MPEG Audio Layer-3** ，简单的说就是一种声音文件的压缩格式。1987 年德国的研究机构 IIS（Institute Integrierte Schaltungen）开始着手一项声音编码及数字音频广播的计划，名称叫做 EUREKA EUl47 ，即 MP3 的前身。之后，这项计划由 IIS 与 Erlangen 大学共同合作，开发出一套非常强大的算法，经由 150 国际标准组织认证之后，符合 ISO-MPEG Audio Layer-3 标准，就成为现在的 MP3。	MPEG 音频编码的层次越高，编码器越复杂，压缩率也越高（MP1 压缩率 4:1 ， MP2 压缩率 6:1 - 8:1 ， MP3 压缩率 10:1 - 12:1）。**有损压缩——感官编码技术**，去除数据中人类感官察觉不到的数据。
AAC	高级音频编码（AdvancedAudio Coding，AAC）一种基于 MPEG-4 的音频编码技术，它由杜比实验室、AT&T 等公司共同研发，目的是替换 MP3 编码方式。作为一种高压缩比的音频压缩算法，AAC 的数据压缩比约为 18:1 ，压缩后的音质可以同未压缩的 CD 音质相媲美。因此，相对于 MP3 、WMA 等音频编码标准来说，在相同质量下码率更低，有效地节约了传输带宽，被广泛得应用于互联网流媒体、IPTV 等领域(低码率，高音质)。 参考 **这篇博客** 。	特点: 1 比特率：AAC- 最高 512kbps（双声道时）/MP3- 32~320kbps 2 采样率：AAC- 最高96kHz / MP3 - 最高48kHz 3 声道数：AAC– 最高48个全音域声道/MP3 - 两声道 4 采样精度：AAC- 最高32bit / MP3 - 最高16bit 5 AAC的不足之处是，它属于有损压缩的格式，相对于APE和FLAC等主流无损压缩，音色“饱满度”差距比较大。另外，除了流媒体网络传输，其所能支持的设备较少。
WAV	微软和 IBM 共同开发的 PC 标准声音格式，文件后缀名 .wav ，是一种通用的音频数据文件。	**通常使用 WAV 格式用来保存一些没有压缩的音频**，也就是经过 PCM 编码后的音频，因此也称为波形文件，依照声音的波形进行存储，因此要占用较大的存储空间。 **WAV 文件也可以存放压缩音频**，但其本身的文件结构使之更加适合于存放原始音频数据并用作进一步的处理。
WMA	Windows Media Audio，微软针对网络音频开发的数位音乐文件格式。	支持防复制，限制播放时间和播放次数甚至播放设备，支持**串流技术**，即**一边读一边播**，轻松实现线上广播。但是算法复杂且封闭。
RA	Real Audio，是 RealNetwork 公司推出的一种流式压缩声音格式，是为了解决网络传输带宽资源限制而设计的。	支持**串流技术**，即**一边读一边播**，轻松实现线上广播。支持使用特殊通讯协议来隐藏音乐文件的真实 URI ，实现仅在线播放不支持下载的功能。

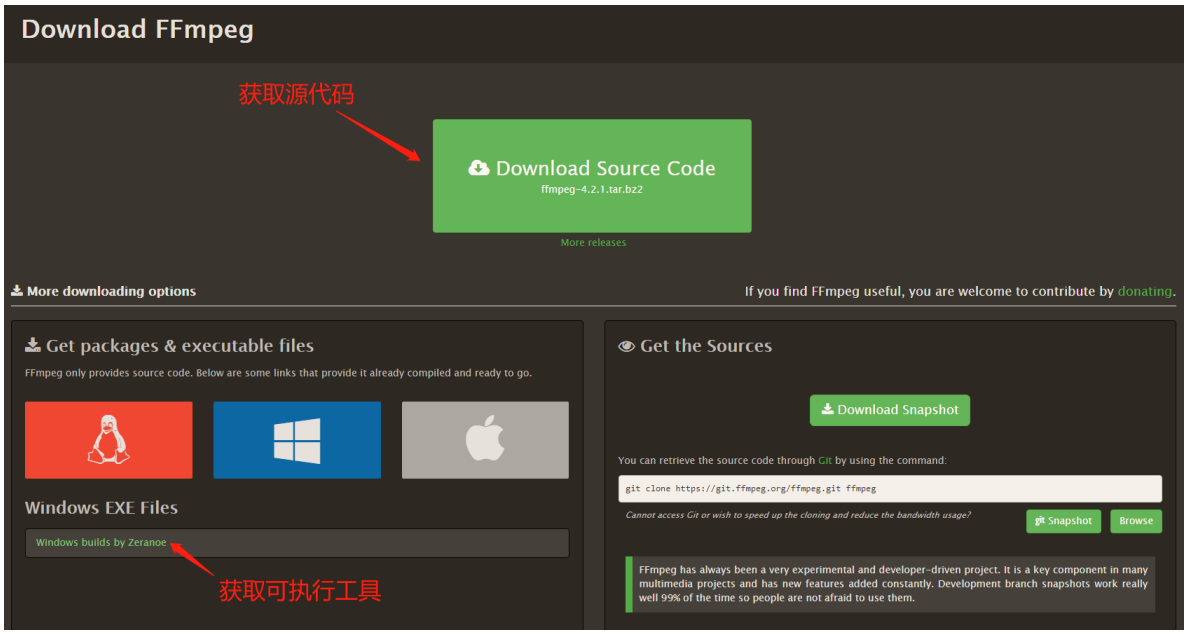
格式	描述	备注
MID	MID 是通过数字化乐器接口 MIDI 输入的声音文件的扩展名，这种文件只是像记乐谱一样地记录下演奏的符号，所以体积是所有音频格式中**最小的**。	数据量小，占用存储空间极小，适合在网络上传输。编辑修改灵活方便，可通过音序器自由的改变MIDI文件的曲调、音色、速度等，甚至可以改换不同的乐器。MIDI 声音仅适于重现打击乐或一些电子乐器的声音（音源窄）。

2.3 音视频封装数据格式

格式	后缀	描述
AVI	.avi	它的英文全称为**Audio Video Interleaved**，即音频视频交错格式。它于1992 年被 Microsoft 公司推出。这种视频格式的优点是图像质量好。由于无损 AVI 可以保存 alpha 通道，经常被我们使用。缺点太多，体积过于庞大，而且更加糟糕的是压缩标准不统一，最普遍的现象就是高版本 Windows 媒体播放器播放不了采用早期编码编辑的 AVI 格式视频，而低版本 Windows 媒体播放器又播放不了采用最新编码编辑的 AVI 格式视频，所以我们在进行一些 AVI 格式的视频播放时常会出现由于视频编码问题而造成的视频不能播放或即使能够播放，但存在不能调节播放进度和播放时只有声音没有图像等一些莫名其妙的问题。
DV-AVI	.avi	DV 的英文全称是 Digital Video Format，是由索尼、松下、JVC 等多家厂商联合提出的一种家用数字视频格式。
QuickTime File Format	.mov	美国 Apple 公司开发的一种视频格式，默认的播放器是苹果的 QuickTime 。具有较高的压缩比率和较完美的视频清晰度等特点，并可以保存 alpha 通道。
MPEG	.mpg .mpeg .mpe .dat .vob .asf .3gp .mp4 等	它的英文全称为 Moving Picture Experts Group，即运动图像专家组格式，该专家组建于 1988 年，专门负责为 CD 建立视频和音频标准，而成员都是为视频、音频及系统领域的技术专家。MPEG文件格式是运动图像压缩算法的国际标准。MPEG 格式目前有三个压缩标准，分别是 MPEG - 1 、MPEG - 2 、和MPEG - 4 。
WMV	.wmv .asf	它的英文全称为 Windows Media Video ，也是微软推出的一种采用独立编码方式并且可以直接在网上实时观看视频节目的文件压缩格式。
Real Video	.rm .rmvb	Real Networks 公司所制定的音频视频压缩规范称为 Real Media 。用户可以使用 RealPlayer 根据不同的网络传输速率制定出不同的压缩比率，从而实现在低速率的网络上进行影像数据实时传送和播放。RMVB 格式：这是一种由 RM 视频格式升级延伸出的新视频格式，当然性能上有很大的提升。RMVB 视频也是有着较明显的优势，一部大小为 700MB 左右的 DVD 影片，如果将其转录成同样品质的 RMVB 格式，其个头最多也就 400MB 左右。大家可能注意到了，以前在网络上下载电影和视频的时候，经常接触到 RMVB 格式，但是随着时代的发展这种格式被越来越多的更优秀的格式替代，著名的人人影视字幕组在2013年已经宣布不再压制RMVB格式视频。
Flash Video	.flv	由 Adobe Flash 延伸出来的的一种流行网络视频封装格式。随着视频网站的丰富，这个格式已经非常普及。
Mastroska	.mkv	是一种新的多媒体封装格式，这个封装格式可把多种不同编码的视频及 16 条或以上不同格式的音频和语言不同的字幕封装到一个 Matroska Media 档内。它也是其中一种开放源代码的多媒体封装格式。 Matroska 同时还可以提供非常好的交互功能，而且比 MPEG 的方便、强大。

2 获取 FFmpeg

- 进入 **FFmpeg 官网** 下载开源代码和可执行工具。



2.1 源代码

FFmpeg 是基于 C 语言的开源项目，其源代码中包含了所有开发人员所需要的包括音视频编解码，音视频播放，音视频处理等 .c/.h 文件，并且源代码根据功能进行了模块划分。主要包含以下几个模块：

模块	简介
libavformat	用于各种音视频封装格式的生成和解析
libavcodec	用于各种类型声音/图像的编解码
libavfilter	音视频滤波器的开发，例如水印
libavutil	包含公共的工具函数
libswresample	原始音频格式转码
libswscale	图像格式转换、缩放等，如 rgb888 转换 yuv420
libpostproc	用于后期效果处理

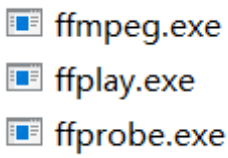
FFmpeg 实际上也是一个引擎，能够集成包括 librtmp、libmp3lame 等第三方的库，以 FFmpeg 统一接口使用。

2.2 可执行工具

可执行工具位于 bin 目录下：

```
E:\dir_where_you_store_ffmpeg_exe\ffmpeg-20191119-0321bde-win64-static\bin
```

将该目录配置到环境变量，我们便可以在 CMD 命令行调用，可执行工具有以下三个：



工具	简介	举例
ffmpeg	对音视频进行各种处理（转码、缩放等）	ffmpeg -i /dir/input.mp4 -s 100x100 /dir/output.mp4
ffplay	播放音视频	ffplay -i /dir/input.mp4
ffprobe	查看音视频文件详细信息	ffprobe -i /dir/input.mp4 ffprobe -i /dir/input.mp4 -v quiet **-print_format json** -show_streams

3 FFmpeg 中几个重要的结构体 简单介绍

结构体一 AVPacket

> 参考博客 **FFmpeg 结构体：AVPacket 解析**

```
// C
typedef struct AVPacket {
    AVBufferRef *buf; //用来管理data指针引用的数据缓存
    int64_t pts; //显示时间，结合AVStream->time_base转换成时间戳
    int64_t dts; //解码时间，结合AVStream->time_base转换成时间戳
    uint8_t *data; //★指向保存压缩数据的指针，这就是AVPacket的实际数据
    int   size; //data的大小
    int   stream_index; //packet在stream的index位置
    int   flags; //标示，结合AV_PKT_FLAG使用，其中最低为1表示该数据是一个关键帧。
    /*
     * flags 可选：
     * #define AV_PKT_FLAG_KEY   0x0001 //关键帧
     * #define AV_PKT_FLAG_CORRUPT 0x0002 //损坏的数据
     * #define AV_PKT_FLAG_DISCARD  0x0004 /丢弃的数据
     */
    AVPacketSideData *side_data; //容器提供的一些附加数据
    int side_data_elems; //边缘数据元数个数
    int64_t duration; //数据的时长，以所属媒体流的时间基准为单位，未知则值为默认值0
    int64_t pos; //数据在流媒体中的位置，未知则值为默认值-1
#if FF_API_CONVERGENCE_DURATION
    attribute_deprecated
    int64_t convergence_duration; //该字段已deprecated,不在使用
#endif
} AVPacket;
```

- AVPacket 结构体**定义在 <libavcodec/avcodec.h>** 中，看源码英文注释去这里找。
- **简介：** AVPacket是FFmpeg中很重要的一个数据结构，它保存了解复用（demuxer)之后，解码（decode）之前的数据（仍然是压缩后的数据）和关于这些数据的一些附加的信息，如显示时间戳（pts），解码时间戳（dts）,数据时长（duration），所在流媒体的索引（stream_index）等等。
- **AVPacket 中的内存管理**请参考上述引用的参考博客。

结构体二 AVFrame

> 参考博客 **FFmpeg 数据结构 AVFrame**

- AVFrame 结构体**定义在 <libavutil/frame.h>**，看源码去这里找，略长不粘。
- **简介：** AVFrame中存储的是经过解码后的原始数据。在解码中，AVFrame是解码器的输出；在编码中，AVFrame是编码器的输入。

结构体三 AVFormatContext

> 参考博客 **FFmpeg 结构体分析：AVFormatContext**

- AVFormatContext 结构体**定义在 <libavformat/avformat.h>** 中，看源码去这里找，略长不粘。
- **简介：**在使用 FFmpeg 进行开发的时候，AVFormatContext 是一个**贯穿始终的数据结构**，很多函数都要用到它作为参数。它是 FFmpeg 解封装（flv，mp4，rmvb，avi）功能的结构体。下面看几个主要变量的作用（在这里考虑解码的情况）：

```
// C
struct AVInputFormat *iformat; //输入数据的封装格式
AVIOContext *pb; //输入数据的缓存
unsigned int nb_streams; //视音频流的个数
AVStream **streams; //视音频流
char filename[1024]; //文件名
int64_t duration; //时长（单位：微秒ms，转换为秒需要除以1_000_000）
int bit_rate; //比特率（单位bps，转换为kbps需要除以1_000）
AVDictionary *metadata; //元数据
```

结构体四 AVCodecContext

> 参考博客 **FFmpeg 结构体分析：AVCodecContext**

- AVCodecContext 结构体**定义在 <libavcodec/avcodec.h>** 中，看源码去这里找，略长不粘。
- **简介：**一些关键的变量来看看（这里只考虑解码）。

```
// C
enum AVMediaType codec_type; //编解码器的类型（视频，音频...）
struct AVCodec  *codec; //采用的解码器AVCodec（H.264,MPEG2...）
int bit_rate; //平均比特率
uint8_t *extradata; int extradata_size; //针对特定编码器包含的附加信息（例如对于H.264解码器来说，存储SPS，PPS等）
AVRational time_base; //根据该参数，可以把PTS转化为实际的时间（单位为秒s）
int width, height; //如果是视频的话，代表宽和高
int refs; //运动估计参考帧的个数（H.264的话会有多帧，MPEG2这类的一般就没有了）
int sample_rate; //采样率（音频）
int channels; //声道数（音频）
enum AVSampleFormat sample_fmt; //采样格式
int profile; //型（H.264里面就有，其他编码标准应该也有）
int level; //级（和profile差不太多）
```

重要结构体之间的关系

> 参考博客 **FFmpeg 中最关键的结构体之间的关系**

FFmpeg 中结构体很多。最关键的结构体可以分成以下几类：

解协议（http, rtsp, rtmp, mms）

AVIOContext ， URLProtocol ， URLContext 主要存储视音频使用的协议的类型以及状态。URLProtocol 存储输入视音频使用的封装格式。每种协议都对应一个 URLProtocol 结构（注意：FFmpeg 中文件也被当做一种协议 “file” ）。

解封装（flv, avi, rmvb, mp4）

AVFormatContext 主要存储视音频封装格式中包含的信息；AVInputFormat 存储输入视音频使用的封装格式。每种视音频封装格式都对应一个 AVInputFormat 结构。

解码（h264, mpeg2, aac, mp3）

每个 AVStream 存储一个视频/音频流的相关数据；每个 AVStream 对应一个 AVCodecContext ， 存储该视频/音频流使用解码方式的相关数据；每个 AVCodecContext 中对应一个 AVCodec ， 包含该视频/音频对应的解码器。每种解码器都对应一个 AVCodec 结构。

存数据

视频的话，每个结构一般是存一帧；音频可能有好几帧

解码前数据：AVPacket

解码后数据：AVFrame