

中科院声学所离线ffmpeg

数据结构设计说明

目录

[1.概述 5](#_Toc7727)

[1.1简介 5](#_Toc7477)

[1.2 读者对象 5](#_Toc6525)

[3. 数据结构设计 6](#_Toc15689)

[3.1AVFrame 6](#_Toc14826)

[3.2AVCodec 11](#_Toc13864)

[3.3 AVStream 13](#_Toc11388)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.0 | 王超锋 |  | 20211117 |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 1.概述

1

1.1简介

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](https://baike.baidu.com/item/GPL/2357903" \t "https://baike.baidu.com/item/ffmpeg/_blank)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。

## 1.2 读者对象

本文档是对SOX代码流程和相关技术点的整体说明，因此读者最好满足如下几点条件后再读此文档：

* 有一定的C和C++语言基础

# 3. 数据结构设计

3

本节主要介绍AVFrame、AVCodec、AVStream结构体定义说明。

### **3.1** AVFrame

AVFrame结构体一般用于存储原始数据（即非压缩数据，例如对视频来说是YUV，RGB，对音频来说是PCM），此外还包含了一些相关的信息。

typedef struct AVFrame {

#define AV\_NUM\_DATA\_POINTERS 8

uint8\_t \*data[AV\_NUM\_DATA\_POINTERS];

int linesize[AV\_NUM\_DATA\_POINTERS];

uint8\_t \*\*extended\_data;

int width, height;

int nb\_samples;

int format;

int key\_frame;

enum AVPictureType pict\_type;

AVRational sample\_aspect\_ratio;

int64\_t pts;

#if FF\_API\_PKT\_PTS

attribute\_deprecated

int64\_t pkt\_pts;

#endif

int64\_t pkt\_dts;

int coded\_picture\_number;

int display\_picture\_number;

int quality;

void \*opaque;

#if FF\_API\_ERROR\_FRAME

attribute\_deprecated

uint64\_t error[AV\_NUM\_DATA\_POINTERS];

#endif

int repeat\_pict;

int interlaced\_frame;

int top\_field\_first;

int palette\_has\_changed;

int64\_t reordered\_opaque;

int sample\_rate;

uint64\_t channel\_layout;

AVBufferRef \*buf[AV\_NUM\_DATA\_POINTERS];

AVBufferRef \*\*extended\_buf;

int nb\_extended\_buf;

AVFrameSideData \*\*side\_data;

int nb\_side\_data;

#define AV\_FRAME\_FLAG\_CORRUPT (1 << 0)

#define AV\_FRAME\_FLAG\_DISCARD (1 << 2)

int flags;

enum AVColorRange color\_range;

enum AVColorPrimaries color\_primaries;

enum AVColorTransferCharacteristic color\_trc;

enum AVColorSpace colorspace;

enum AVChromaLocation chroma\_location;

int64\_t best\_effort\_timestamp;

int64\_t pkt\_pos;

int64\_t pkt\_duration;

AVDictionary \*metadata;

int decode\_error\_flags;

#define FF\_DECODE\_ERROR\_INVALID\_BITSTREAM 1

#define FF\_DECODE\_ERROR\_MISSING\_REFERENCE 2

#define FF\_DECODE\_ERROR\_CONCEALMENT\_ACTIVE 4

#define FF\_DECODE\_ERROR\_DECODE\_SLICES 8

int channels;

int pkt\_size;

#if FF\_API\_FRAME\_QP

attribute\_deprecated

int8\_t \*qscale\_table;

attribute\_deprecated

int qstride;

attribute\_deprecated

int qscale\_type;

attribute\_deprecated

AVBufferRef \*qp\_table\_buf;

#endif

AVBufferRef \*hw\_frames\_ctx;

AVBufferRef \*opaque\_ref;

size\_t crop\_top;

size\_t crop\_bottom;

size\_t crop\_left;

size\_t crop\_right;

AVBufferRef \*private\_ref;

} AVFrame;

重要变量说明：

uint8\_t \*data[AV\_NUM\_DATA\_POINTERS]：解码后原始数据（对视频来说是YUV，RGB，对音频来说是PCM）

int linesize[AV\_NUM\_DATA\_POINTERS]：data中“一行”数据的大小。注意：未必等于图像的宽，一般大于图像的宽。

int width, height：视频帧宽和高（1920x1080,1280x720...）

int nb\_samples：音频的一个AVFrame中可能包含多个音频帧，在此标记包含了几个

int format：解码后原始数据类型（YUV420，YUV422，RGB24...）

int key\_frame：是否是关键帧

enum AVPictureType pict\_type：帧类型（I,B,P...）

AVRational sample\_aspect\_ratio：宽高比（16:9，4:3...）

int64\_t pts：显示时间戳

int coded\_picture\_number：编码帧序号

int display\_picture\_number：显示帧序号

int8\_t \*qscale\_table：QP表

uint8\_t \*mbskip\_table：跳过宏块表

int16\_t (\*motion\_val[2])[2]：运动矢量表

uint32\_t \*mb\_type：宏块类型表

short \*dct\_coeff：DCT系数，这个没有提取过

int8\_t \*ref\_index[2]：运动估计参考帧列表（貌似H.264这种比较新的标准才会涉及到多参考帧）

int interlaced\_frame：是否是隔行扫描

uint8\_t motion\_subsample\_log2：一个宏块中的运动矢量采样个数，取log的

### **3.2** AVCodec

AVCodec是存储编解码器信息的结构体。

typedef struct AVCodec {

const char \*name;

const char \*long\_name;

enum AVMediaType type;

enum AVCodecID id;

int capabilities;

const AVRational \*supported\_framerates; ///< array of supported framerates, or NULL if any, array is terminated by {0,0}

const enum AVPixelFormat \*pix\_fmts; ///< array of supported pixel formats, or NULL if unknown, array is terminated by -1

const int \*supported\_samplerates; ///< array of supported audio samplerates, or NULL if unknown, array is terminated by 0

const enum AVSampleFormat \*sample\_fmts; ///< array of supported sample formats, or NULL if unknown, array is terminated by -1

const uint64\_t \*channel\_layouts; ///< array of support channel layouts, or NULL if unknown. array is terminated by 0

uint8\_t max\_lowres; ///< maximum value for lowres supported by the decoder

const AVClass \*priv\_class; ///< AVClass for the private context

const AVProfile \*profiles; ///< array of recognized profiles, or NULL if unknown, array is terminated by {FF\_PROFILE\_UNKNOWN}

const char \*wrapper\_name;

int priv\_data\_size;

struct AVCodec \*next;

int (\*update\_thread\_context)(struct AVCodecContext \*dst, const struct AVCodecContext \*src);

const AVCodecDefault \*defaults;

void (\*init\_static\_data)(struct AVCodec \*codec);

int (\*init)(struct AVCodecContext \*);

int (\*encode\_sub)(struct AVCodecContext \*, uint8\_t \*buf, int buf\_size,

const struct AVSubtitle \*sub);

int (\*encode2)(struct AVCodecContext \*avctx, struct AVPacket \*avpkt,

const struct AVFrame \*frame, int \*got\_packet\_ptr);

int (\*decode)(struct AVCodecContext \*, void \*outdata, int \*outdata\_size, struct AVPacket \*avpkt);

int (\*close)(struct AVCodecContext \*);

int (\*send\_frame)(struct AVCodecContext \*avctx, const struct AVFrame \*frame);

int (\*receive\_packet)(struct AVCodecContext \*avctx, struct AVPacket \*avpkt);

int (\*receive\_frame)(struct AVCodecContext \*avctx, struct AVFrame \*frame);

void (\*flush)(struct AVCodecContext \*);

int caps\_internal;

const char \*bsfs;

const struct AVCodecHWConfigInternal \*\*hw\_configs;

const uint32\_t \*codec\_tags;

} AVCodec;

重要变量说明：

const char \*name：编解码器的名字，比较短

const char \*long\_name：编解码器的名字，全称，比较长

enum AVMediaType type：指明了类型，是视频，音频，还是字幕

enum AVCodecID id：ID，不重复

const AVRational \*supported\_framerates：支持的帧率（仅视频）

const enum AVPixelFormat \*pix\_fmts：支持的像素格式（仅视频）

const int \*supported\_samplerates：支持的采样率（仅音频）

const enum AVSampleFormat \*sample\_fmts：支持的采样格式（仅音频）

const uint64\_t \*channel\_layouts：支持的声道数（仅音频）

int priv\_data\_size：私有数据的大小

### 3.3 AVStream

AVStream是存储每一个视频/音频流信息的结构体。

typedef struct AVStream {

int index; /\*\*< stream index in AVFormatContext \*/

int id;

#if FF\_API\_LAVF\_AVCTX

attribute\_deprecated

AVCodecContext \*codec;

#endif

void \*priv\_data;

AVRational time\_base;

int64\_t start\_time;

int64\_t duration;

int64\_t nb\_frames; ///< number of frames in this stream if known or 0

int disposition; /\*\*< AV\_DISPOSITION\_\* bit field \*/

enum AVDiscard discard; ///< Selects which packets can be discarded at will and do not need to be demuxed.

AVRational sample\_aspect\_ratio;

AVDictionary \*metadata;

AVRational avg\_frame\_rate;

AVPacket attached\_pic;

AVPacketSideData \*side\_data;

int nb\_side\_data;

int event\_flags;

#define AVSTREAM\_EVENT\_FLAG\_METADATA\_UPDATED 0x0001 ///< The call resulted in updated metadata.

AVRational r\_frame\_rate;

#if FF\_API\_LAVF\_FFSERVER

attribute\_deprecated

char \*recommended\_encoder\_configuration;

#endif

AVCodecParameters \*codecpar;

#define MAX\_STD\_TIMEBASES (30\*12+30+3+6)/

struct {

int64\_t last\_dts;

int64\_t duration\_gcd;

int duration\_count;

int64\_t rfps\_duration\_sum;

double (\*duration\_error)[2][MAX\_STD\_TIMEBASES];

int64\_t codec\_info\_duration;

int64\_t codec\_info\_duration\_fields;

int frame\_delay\_evidence;

int found\_decoder;

int64\_t last\_duration;

int64\_t fps\_first\_dts;

int fps\_first\_dts\_idx;

int64\_t fps\_last\_dts;

int fps\_last\_dts\_idx;

} \*info;

int pts\_wrap\_bits; /\*\*< number of bits in pts (used for wrapping control) \*/

int64\_t first\_dts;

int64\_t cur\_dts;

int64\_t last\_IP\_pts;

int last\_IP\_duration;

int probe\_packets;

int codec\_info\_nb\_frames;

enum AVStreamParseType need\_parsing;

struct AVCodecParserContext \*parser;

struct AVPacketList \*last\_in\_packet\_buffer;

AVProbeData probe\_data;

#define MAX\_REORDER\_DELAY 16

int64\_t pts\_buffer[MAX\_REORDER\_DELAY+1];

AVIndexEntry \*index\_entries; /\*\*< Only used if the format does not support seeking natively. \*/

int nb\_index\_entries;

unsigned int index\_entries\_allocated\_size;

int stream\_identifier;

int program\_num;

int pmt\_version;

int pmt\_stream\_idx;

int64\_t interleaver\_chunk\_size;

int64\_t interleaver\_chunk\_duration;

int request\_probe;

int skip\_to\_keyframe;

int skip\_samples;

int64\_t start\_skip\_samples;

int64\_t first\_discard\_sample;

int64\_t last\_discard\_sample;

int nb\_decoded\_frames;

int64\_t mux\_ts\_offset;

int64\_t pts\_wrap\_reference;

int pts\_wrap\_behavior;

int update\_initial\_durations\_done;

int64\_t pts\_reorder\_error[MAX\_REORDER\_DELAY+1];

uint8\_t pts\_reorder\_error\_count[MAX\_REORDER\_DELAY+1];

int64\_t last\_dts\_for\_order\_check;

uint8\_t dts\_ordered;

uint8\_t dts\_misordered;

int inject\_global\_side\_data;

AVRational display\_aspect\_ratio;

AVStreamInternal \*internal;

} AVStream;

重要变量说明：

int index：标识该视频/音频流

AVCodecContext \*codec：指向该视频/音频流的AVCodecContext（它们是一一对应的关系）

AVRational time\_base：时基。通过该值可以把PTS，DTS转化为真正的时间。

int64\_t duration：该视频/音频流长度

AVDictionary \*metadata：元数据信息

AVRational avg\_frame\_rate：帧率（注：对视频来说，这个挺重要的）

AVPacket attached\_pic：附带的图片。