原 FFMPEG结构体分析：AVCodecContext

2013年11月08日 00:49:27 阅读数：51202

注：写了一系列的结构体的分析的文章，在这里列一个列表：

FFMPEG结构体分析：AVFrame

FFMPEG结构体分析：AVFormatContext

FFMPEG结构体分析：AVCodecContext

FFMPEG结构体分析：AVIOContext

FFMPEG结构体分析：AVCodec

FFMPEG结构体分析：AVStream

FFMPEG结构体分析：AVPacket

FFMPEG有几个最重要的结构体，包含了解协议，解封装，解码操作，此前已经进行过分析：

FFMPEG中最关键的结构体之间的关系

在此不再详述，其中AVCodecContext是包含变量较多的结构体（感觉差不多是变量最多的结构体）。本文将会大概分析一下该结构体里每个变量的含义和作用。因为如果每个变量都分析的话，工作量太大，实在来不及。

首先看一下结构体的定义（位于avcodec.h）：

[cpp]

```cpp
/*
 *雷霄骅
 *leixiaohua1020@126.com
 *中国传媒大学/数字电视技术
 */
/**
 * main external API structure.
 * New fields can be added to the end with minor version bumps.
 * Removal, reordering and changes to existing fields require a major
 * version bump.
 * Please use AVOptions (av_opt* / av_set/get*()) to access these fields from user
 * applications.
 * sizeof(AVCodecContext) must not be used outside libav*.
 */
typedef struct AVCodecContext {
    /**
     * information on struct for av_log
     * - set by avcodec_alloc_context3
     */
    const AVClass *av_class;
    int log_level_offset;

    enum AVMediaType codec_type; /* see AVMEDIA_TYPE_xxx */
    const struct AVCodec  *codec;
    char             codec_name[32];
    enum AVCodecID     codec_id; /* see AV_CODEC_ID_xxx */

    /**
     * fourcc (LSB first, so "ABCD" -> ('D'<<24) + ('C'<<16) + ('B'<<8) + 'A').
     * This is used to work around some encoder bugs.
     * A demuxer should set this to what is stored in the field used to identify the codec.
     * If there are multiple such fields in a container then the demuxer should choose the one
     * which maximizes the information about the used codec.
     * If the codec tag field in a container is larger than 32 bits then the demuxer should
     * remap the longer ID to 32 bits with a table or other structure. Alternatively a new
     * extra_codec_tag + size could be added but for this a clear advantage must be demonstrated
     * first.
     * - encoding: Set by user, if not then the default based on codec_id will be used.
     * - decoding: Set by user, will be converted to uppercase by libavcodec during init.
     */
    unsigned int codec_tag;

    /**
     * fourcc from the AVI stream header (LSB first, so "ABCD" -> ('D'<<24) + ('C'<<16) + ('B'<<8) + 'A').
     * This is used to work around some encoder bugs.
     * - encoding: unused
     * - decoding: Set by user, will be converted to uppercase by libavcodec during init.
     */
    unsigned int stream_codec_tag;

#if FF_API_SUB_ID
    /**
```

```
     * @deprecated this field is unused
     */
    attribute_deprecated int sub_id;
#endif

    void *priv_data;

    /**
     * Private context used for internal data.
     *
     * Unlike priv_data, this is not codec-specific. It is used in general
     * libavcodec functions.
     */
    struct AVCodecInternal *internal;

    /**
     * Private data of the user, can be used to carry app specific stuff.
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    void *opaque;

    /**
     * the average bitrate
     * - encoding: Set by user; unused for constant quantizer encoding.
     * - decoding: Set by libavcodec. 0 or some bitrate if this info is available in the stream.
     */
    int bit_rate;

    /**
     * number of bits the bitstream is allowed to diverge from the reference.
     *           the reference can be CBR (for CBR pass1) or VBR (for pass2)
     * - encoding: Set by user; unused for constant quantizer encoding.
     * - decoding: unused
     */
    int bit_rate_tolerance;

    /**
     * Global quality for codecs which cannot change it per frame.
     * This should be proportional to MPEG-1/2/4 qscale.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int global_quality;

    /**
     * - encoding: Set by user.
     * - decoding: unused
     */
    int compression_level;
#define FF_COMPRESSION_DEFAULT -1

    /**
     * CODEC_FLAG_*.
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int flags;

    /**
     * CODEC_FLAG2_*
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int flags2;

    /**
     * some codecs need / can use extradata like Huffman tables.
     * mjpeg: Huffman tables
     * rv10: additional flags
     * mpeg4: global headers (they can be in the bitstream or here)
     * The allocated memory should be FF_INPUT_BUFFER_PADDING_SIZE bytes larger
     * than extradata_size to avoid prolems if it is read with the bitstream reader.
     * The bytewise contents of extradata must not depend on the architecture or CPU endianness.
     * - encoding: Set/allocated/freed by libavcodec.
     * - decoding: Set/allocated/freed by user.
     */
    uint8_t *extradata;
    int extradata_size;

    /**
     * This is the fundamental unit of time (in seconds) in terms
     * of which frame timestamps are represented. For fixed-fps content,
     * timebase should be 1/framerate and timestamp increments should be
     * identically 1.
     * - encoding: MUST be set by user.
     * - decoding: Set by libavcodec.
     */
    AVRational time_base;

    /**
```

```
     * For some codecs, the time base is closer to the field rate than the frame rate.
     * Most notably, H.264 and MPEG-2 specify time_base as half of frame duration
     * if no telecine is used ...
     *
     * Set to time_base ticks per frame. Default 1, e.g., H.264/MPEG-2 set it to 2.
     */
    int ticks_per_frame;

    /**
     * Encoding: Number of frames delay there will be from the encoder input to
     *           the decoder output. (we assume the decoder matches the spec)
     * Decoding: Number of frames delay in addition to what a standard decoder
     *           as specified in the spec would produce.
     *
     * Video:
     *   Number of frames the decoded output will be delayed relative to the
     *   encoded input.
     *
     * Audio:
     *   For encoding, this is the number of "priming" samples added to the
     *   beginning of the stream. The decoded output will be delayed by this
     *   many samples relative to the input to the encoder. Note that this
     *   field is purely informational and does not directly affect the pts
     *   output by the encoder, which should always be based on the actual
     *   presentation time, including any delay.
     *   For decoding, this is the number of samples the decoder needs to
     *   output before the decoder's output is valid. When seeking, you should
     *   start decoding this many samples prior to your desired seek point.
     *
     * - encoding: Set by libavcodec.
     * - decoding: Set by libavcodec.
     */
    int delay;


    /* video only */
    /**
     * picture width / height.
     * - encoding: MUST be set by user.
     * - decoding: Set by libavcodec.
     * Note: For compatibility it is possible to set this instead of
     * coded_width/height before decoding.
     */
    int width, height;

    /**
     * Bitstream width / height, may be different from width/height if lowres enabled.
     * - encoding: unused
     * - decoding: Set by user before init if known. Codec should override / dynamically change if needed.
     */
    int coded_width, coded_height;

#define FF_ASPECT_EXTENDED 15

    /**
     * the number of pictures in a group of pictures, or 0 for intra_only
     * - encoding: Set by user.
     * - decoding: unused
     */
    int gop_size;

    /**
     * Pixel format, see AV_PIX_FMT_xxx.
     * May be set by the demuxer if known from headers.
     * May be overridden by the decoder if it knows better.
     * - encoding: Set by user.
     * - decoding: Set by user if known, overridden by libavcodec if known
     */
    enum AVPixelFormat pix_fmt;

    /**
     * Motion estimation algorithm used for video coding.
     * 1 (zero), 2 (full), 3 (log), 4 (phods), 5 (epzs), 6 (x1), 7 (hex),
     * 8 (umh), 9 (iter), 10 (tesa) [7, 8, 10 are x264 specific, 9 is snow specific]
     * - encoding: MUST be set by user.
     * - decoding: unused
     */
    int me_method;

    /**
     * If non NULL, 'draw_horiz_band' is called by the libavcodec
     * decoder to draw a horizontal band. It improves cache usage. Not
     * all codecs can do that. You must check the codec capabilities
     * beforehand.
     * When multithreading is used, it may be called from multiple threads
     * at the same time; threads might draw different parts of the same AVFrame,
     * or multiple AVFrames, and there is no guarantee that slices will be drawn
     * in order.
     * The function is also used by hardware acceleration APIs.
     * It is called at least once during frame decoding to pass
     * the data needed for hardware render.
     * In that mode instead of pixel data, AVFrame points to
     * a structure specific to the acceleration API. The application
     * reads the structure and can change some fields to indicate progress
     * or mark state.
     * - encoding: unused
     * - decoding: Set by user.
     * @param height the height of the slice
     * @param y the y position of the slice
     * @param type 1->top field, 2->bottom field, 3->frame
     * @param offset offset into the AVFrame.data from which the slice should be read
     */
    void (*draw_horiz_band)(struct AVCodecContext *s,
                            const AVFrame *src, int offset[AV_NUM_DATA_POINTERS],
                            int y, int type, int height);

    /**
     * callback to negotiate the pixelFormat
     * @param fmt is the list of formats which are supported by the codec,
     * it is terminated by -1 as 0 is a valid format, the formats are ordered by quality.
     * The first is always the native one.
     * @return the chosen format
     * - encoding: unused
     * - decoding: Set by user, if not set the native format will be chosen.
     */
    enum AVPixelFormat (*get_format)(struct AVCodecContext *s, const enum AVPixelFormat * fmt);

    /**
     * maximum number of B-frames between non-B-frames
     * Note: The output will be delayed by max_b_frames+1 relative to the input.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int max_b_frames;

    /**
     * qscale factor between IP and B-frames
     * If > 0 then the last P-frame quantizer will be used (q= lastp_q*factor+offset).
     * If < 0 then normal ratecontrol will be done (q= -normal_q*factor+offset).
     * - encoding: Set by user.
     * - decoding: unused
     */
    float b_quant_factor;

    /** obsolete FIXME remove */
    int rc_strategy;
#define FF_RC_STRATEGY_XVID 1

    int b_frame_strategy;

#if FF_API_MPV_GLOBAL_OPTS
    /**
     * luma single coefficient elimination threshold
     * - encoding: Set by user.
     * - decoding: unused
     */
    attribute_deprecated int luma_elim_threshold;

    /**
     * chroma single coeff elimination threshold
     * - encoding: Set by user.
     * - decoding: unused
     */
    attribute_deprecated int chroma_elim_threshold;
#endif

    /**
     * qscale offset between IP and B-frames
     * - encoding: Set by user.
     * - decoding: unused
     */
    float b_quant_offset;

    /**
     * Size of the frame reordering buffer in the decoder.
     * For MPEG-2 it is 1 IPB or 0 low delay IP.
     * - encoding: Set by libavcodec.
     * - decoding: Set by libavcodec.
     */
    int has_b_frames;

    /**
     * 0-> h263 quant 1-> mpeg quant
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mpeg_quant;

    /**
     * qscale factor between P and I-frames
     * If > 0 then the last p frame quantizer will be used (q= lastp_q*factor+offset).
     * If < 0 then normal ratecontrol will be done (q= -normal_q*factor+offset).
     * - encoding: Set by user.
     * - decoding: unused
     */
    float i_quant_factor;

    /**
     * qscale offset between P and I-frames
     * - encoding: Set by user.
     * - decoding: unused
     */
    float i_quant_offset;

    /**
     * luminance masking (0-> disabled)
     * - encoding: Set by user.
     * - decoding: unused
     */
    float lumi_masking;

    /**
     * temporary complexity masking (0-> disabled)
     * - encoding: Set by user.
     * - decoding: unused
     */
    float temporal_cplx_masking;

    /**
     * spatial complexity masking (0-> disabled)
     * - encoding: Set by user.
     * - decoding: unused
     */
    float spatial_cplx_masking;

    /**
     * p block masking (0-> disabled)
     * - encoding: Set by user.
     * - decoding: unused
     */
    float p_masking;

    /**
     * darkness masking (0-> disabled)
     * - encoding: Set by user.
     * - decoding: unused
     */
    float dark_masking;

    /**
     * slice count
     * - encoding: Set by libavcodec.
     * - decoding: Set by user (or 0).
     */
    int slice_count;
    /**
     * prediction method (needed for huffyuv)
     * - encoding: Set by user.
     * - decoding: unused
     */
     int prediction_method;
#define FF_PRED_LEFT   0
#define FF_PRED_PLANE  1
#define FF_PRED_MEDIAN 2

    /**
     * slice offsets in the frame in bytes
     * - encoding: Set/allocated by libavcodec.
     * - decoding: Set/allocated by user (or NULL).
     */
    int *slice_offset;

    /**
     * sample aspect ratio (0 if unknown)
     * That is the width of a pixel divided by the height of the pixel.
     * Numerator and denominator must be relatively prime and smaller than 256 for some video standards.
     * - encoding: Set by user.
     * - decoding: Set by libavcodec.
     */
    AVRational sample_aspect_ratio;

    /**
     * motion estimation comparison function
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_cmp;
    /**
     * subpixel motion estimation comparison function
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_sub_cmp;
    /**
```

```c
    /**
     * macroblock comparison function (not supported yet)
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mb_cmp;
    /**
     * interlaced DCT comparison function
     * - encoding: Set by user.
     * - decoding: unused
     */
    int ildct_cmp;
#define FF_CMP_SAD    0
#define FF_CMP_SSE    1
#define FF_CMP_SATD   2
#define FF_CMP_DCT    3
#define FF_CMP_PSNR   4
#define FF_CMP_BIT    5
#define FF_CMP_RD     6
#define FF_CMP_ZERO   7
#define FF_CMP_VSAD   8
#define FF_CMP_VSSE   9
#define FF_CMP_NSSE   10
#define FF_CMP_W53    11
#define FF_CMP_W97    12
#define FF_CMP_DCTMAX 13
#define FF_CMP_DCT264 14
#define FF_CMP_CHROMA 256

    /**
     * ME diamond size & shape
     * - encoding: Set by user.
     * - decoding: unused
     */
    int dia_size;

    /**
     * amount of previous MV predictors (2a+1 x 2a+1 square)
     * - encoding: Set by user.
     * - decoding: unused
     */
    int last_predictor_count;

    /**
     * prepass for motion estimation
     * - encoding: Set by user.
     * - decoding: unused
     */
    int pre_me;

    /**
     * motion estimation prepass comparison function
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_pre_cmp;

    /**
     * ME prepass diamond size & shape
     * - encoding: Set by user.
     * - decoding: unused
     */
    int pre_dia_size;

    /**
     * subpel ME quality
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_subpel_quality;

    /**
     * DTG active format information (additional aspect ratio
     * information only used in DVB MPEG-2 transport streams)
     * 0 if not set.
     *
     * - encoding: unused
     * - decoding: Set by decoder.
     */
    int dtg_active_format;
#define FF_DTG_AFD_SAME         8
#define FF_DTG_AFD_4_3          9
#define FF_DTG_AFD_16_9         10
#define FF_DTG_AFD_14_9         11
#define FF_DTG_AFD_4_3_SP_14_9  13
#define FF_DTG_AFD_16_9_SP_14_9 14
#define FF_DTG_AFD_SP_4_3       15

    /**
     * maximum motion estimation search range in subpel units
     * If 0 then no limit.
     *
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_range;

    /**
     * intra quantizer bias
     * - encoding: Set by user.
     * - decoding: unused
     */
    int intra_quant_bias;
#define FF_DEFAULT_QUANT_BIAS 999999

    /**
     * inter quantizer bias
     * - encoding: Set by user.
     * - decoding: unused
     */
    int inter_quant_bias;

#if FF_API_COLOR_TABLE_ID
    /**
     * color table ID
     * - encoding: unused
     * - decoding: Which clrtable should be used for 8bit RGB images.
     *             Tables have to be stored somewhere. FIXME
     */
    attribute_deprecated int color_table_id;
#endif

    /**
     * slice flags
     * - encoding: unused
     * - decoding: Set by user.
     */
    int slice_flags;
#define SLICE_FLAG_CODED_ORDER    0x0001 ///< draw_horiz_band() is called in coded order instead of display
#define SLICE_FLAG_ALLOW_FIELD    0x0002 ///< allow draw_horiz_band() with field slices (MPEG2 field pics)
#define SLICE_FLAG_ALLOW_PLANE    0x0004 ///< allow draw_horiz_band() with 1 component at a time (SVQ1)

    /**
     * XVideo Motion Acceleration
     * - encoding: forbidden
     * - decoding: set by decoder
     */
    int xvmc_acceleration;

    /**
     * macroblock decision mode
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mb_decision;
#define FF_MB_DECISION_SIMPLE 0        ///< uses mb_cmp
#define FF_MB_DECISION_BITS   1        ///< chooses the one which needs the fewest bits
#define FF_MB_DECISION_RD     2        ///< rate distortion

    /**
     * custom intra quantization matrix
     * - encoding: Set by user, can be NULL.
     * - decoding: Set by libavcodec.
     */
    uint16_t *intra_matrix;

    /**
     * custom inter quantization matrix
     * - encoding: Set by user, can be NULL.
     * - decoding: Set by libavcodec.
     */
    uint16_t *inter_matrix;

    /**
     * scene change detection threshold
     * 0 is default, larger means fewer detected scene changes.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int scenechange_threshold;

    /**
     * noise reduction strength
     * - encoding: Set by user.
     * - decoding: unused
     */
    int noise_reduction;

#if FF_API_INTER_THRESHOLD
    /**
     * @deprecated this field is unused
     */
    attribute_deprecated int inter_threshold;
```

```
#endif

#if FF_API_MPV_GLOBAL_OPTS
    /**
     * @deprecated use mpegvideo private options instead
     */
    attribute_deprecated int quantizer_noise_shaping;
#endif

    /**
     * Motion estimation threshold below which no motion estimation is
     * performed, but instead the user specified motion vectors are used.
     *
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_threshold;

    /**
     * Macroblock threshold below which the user specified macroblock types will be used.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mb_threshold;

    /**
     * precision of the intra DC coefficient - 8
     * - encoding: Set by user.
     * - decoding: unused
     */
    int intra_dc_precision;

    /**
     * Number of macroblock rows at the top which are skipped.
     * - encoding: unused
     * - decoding: Set by user.
     */
    int skip_top;

    /**
     * Number of macroblock rows at the bottom which are skipped.
     * - encoding: unused
     * - decoding: Set by user.
     */
    int skip_bottom;

    /**
     * Border processing masking, raises the quantizer for mbs on the borders
     * of the picture.
     * - encoding: Set by user.
     * - decoding: unused
     */
    float border_masking;

    /**
     * minimum MB lagrange multipler
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mb_lmin;

    /**
     * maximum MB lagrange multipler
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mb_lmax;

    /**
     *
     * - encoding: Set by user.
     * - decoding: unused
     */
    int me_penalty_compensation;

    /**
     *
     * - encoding: Set by user.
     * - decoding: unused
     */
    int bidir_refine;

    /**
     *
     * - encoding: Set by user.
     * - decoding: unused
     */
    int brd_scale;

    /**
     * minimum GOP size
     * - encoding: Set by user.
     * - decoding: unused
     */
    int keyint_min;

    /**
     * number of reference frames
     * - encoding: Set by user.
     * - decoding: Set by lavc.
     */
    int refs;

    /**
     * chroma qp offset from luma
     * - encoding: Set by user.
     * - decoding: unused
     */
    int chromaoffset;

    /**
     * Multiplied by qscale for each frame and added to scene_change_score.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int scenechange_factor;

    /**
     *
     * Note: Value depends upon the compare function used for fullpel ME.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int mv0_threshold;

    /**
     * Adjust sensitivity of b_frame_strategy 1.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int b_sensitivity;

    /**
     * Chromaticity coordinates of the source primaries.
     * - encoding: Set by user
     * - decoding: Set by libavcodec
     */
    enum AVColorPrimaries color_primaries;

    /**
     * Color Transfer Characteristic.
     * - encoding: Set by user
     * - decoding: Set by libavcodec
     */
    enum AVColorTransferCharacteristic color_trc;

    /**
     * YUV colorspace type.
     * - encoding: Set by user
     * - decoding: Set by libavcodec
     */
    enum AVColorSpace colorspace;

    /**
     * MPEG vs JPEG YUV range.
     * - encoding: Set by user
     * - decoding: Set by libavcodec
     */
    enum AVColorRange color_range;

    /**
     * This defines the location of chroma samples.
     * - encoding: Set by user
     * - decoding: Set by libavcodec
     */
    enum AVChromaLocation chroma_sample_location;

    /**
     * Number of slices.
     * Indicates number of picture subdivisions. Used for parallelized
     * decoding.
     * - encoding: Set by user
     * - decoding: unused
     */
    int slices;

    /** Field order
     * - encoding: set by libavcodec
     * - decoding: Set by user.
     */
    enum AVFieldOrder field_order;

    /* audio only */
    int sample_rate; ///< samples per second
    int channels;    ///< number of audio channels

    /**
     * audio sample format
     * - encoding: Set by user.
     * - decoding: Set by libavcodec.
     */
    enum AVSampleFormat sample_fmt;  ///< sample format

    /* The following data should not be initialized. */
    /**
     * Samples per packet, initialized when calling 'init'.
     */
    int frame_size;

    /**
     * Frame counter, set by libavcodec.
     *
     * - decoding: total number of frames returned from the decoder so far.
     * - encoding: total number of frames passed to the encoder so far.
     *
     *   @note the counter is not incremented if encoding/decoding resulted in
     *   an error.
     */
    int frame_number;

    /**
     * number of bytes per packet if constant and known or 0
     * Used by some WAV based audio codecs.
     */
    int block_align;

    /**
     * Audio cutoff bandwidth (0 means "automatic")
     * - encoding: Set by user.
     * - decoding: unused
     */
    int cutoff;

#if FF_API_REQUEST_CHANNELS
    /**
     * Decoder should decode to this many channels if it can (0 for default)
     * - encoding: unused
     * - decoding: Set by user.
     * @deprecated Deprecated in favor of request_channel_layout.
     */
    int request_channels;
#endif

    /**
     * Audio channel layout.
     * - encoding: set by user.
     * - decoding: set by user, may be overwritten by libavcodec.
     */
    uint64_t channel_layout;

    /**
     * Request decoder to use this channel layout if it can (0 for default)
     * - encoding: unused
     * - decoding: Set by user.
     */
    uint64_t request_channel_layout;

    /**
     * Type of service that the audio stream conveys.
     * - encoding: Set by user.
     * - decoding: Set by libavcodec.
     */
    enum AVAudioServiceType audio_service_type;

    /**
     * desired sample format
     * - encoding: Not used.
     * - decoding: Set by user.
     * Decoder will decode to this format if it can.
     */
    enum AVSampleFormat request_sample_fmt;

    /**
     * Called at the beginning of each frame to get a buffer for it.
     *
     * The function will set AVFrame.data[], AVFrame.linesize[].
     * AVFrame.extended_data[] must also be set, but it should be the same as
     * AVFrame.data[] except for planar audio with more channels than can fit
     * in AVFrame.data[]. In that case, AVFrame.data[] shall still contain as
     * many data pointers as it can hold.
     *
     * if CODEC_CAP_DR1 is not set then get_buffer() must call
     * avcodec_default_get_buffer() instead of providing buffers allocated by
```

```c
     * some other means.
     *
     * AVFrame.data[] should be 32- or 16-byte-aligned unless the CPU doesn't
     * need it. avcodec_default_get_buffer() aligns the output buffer properly,
     * but if get_buffer() is overridden then alignment considerations should
     * be taken into account.
     *
     * @see avcodec_default_get_buffer()
     *
     * Video:
     *
     * If pic.reference is set then the frame will be read later by libavcodec.
     * avcodec_align_dimensions2() should be used to find the required width and
     * height, as they normally need to be rounded up to the next multiple of 16.
     *
     * If frame multithreading is used and thread_safe_callbacks is set,
     * it may be called from a different thread, but not from more than one at
     * once. Does not need to be reentrant.
     *
     * @see release_buffer(), reget_buffer()
     * @see avcodec_align_dimensions2()
     *
     * Audio:
     *
     * Decoders request a buffer of a particular size by setting
     * AVFrame.nb_samples prior to calling get_buffer(). The decoder may,
     * however, utilize only part of the buffer by setting AVFrame.nb_samples
     * to a smaller value in the output frame.
     *
     * Decoders cannot use the buffer after returning from
     * avcodec_decode_audio4(), so they will not call release_buffer(), as it
     * is assumed to be released immediately upon return.
     *
     * As a convenience, av_samples_get_buffer_size() and
     * av_samples_fill_arrays() in libavutil may be used by custom get_buffer()
     * functions to find the required data size and to fill data pointers and
     * linesize. In AVFrame.linesize, only linesize[0] may be set for audio
     * since all planes must be the same size.
     *
     * @see av_samples_get_buffer_size(), av_samples_fill_arrays()
     *
     * - encoding: unused
     * - decoding: Set by libavcodec, user can override.
     */
    int (*get_buffer)(struct AVCodecContext *c, AVFrame *pic);

    /**
     * Called to release buffers which were allocated with get_buffer.
     * A released buffer can be reused in get_buffer().
     * pic.data[*] must be set to NULL.
     * May be called from a different thread if frame multithreading is used,
     * but not by more than one thread at once, so does not need to be reentrant.
     * - encoding: unused
     * - decoding: Set by libavcodec, user can override.
     */
    void (*release_buffer)(struct AVCodecContext *c, AVFrame *pic);

    /**
     * Called at the beginning of a frame to get cr buffer for it.
     * Buffer type (size, hints) must be the same. libavcodec won't check it.
     * libavcodec will pass previous buffer in pic, function should return
     * same buffer or new buffer with old frame "painted" into it.
     * If pic.data[0] == NULL must behave like get_buffer().
     * if CODEC_CAP_DR1 is not set then reget_buffer() must call
     * avcodec_default_reget_buffer() instead of providing buffers allocated by
     * some other means.
     * - encoding: unused
     * - decoding: Set by libavcodec, user can override.
     */
    int (*reget_buffer)(struct AVCodecContext *c, AVFrame *pic);


    /* - encoding parameters */
    float qcompress;  ///< amount of qscale change between easy & hard scenes (0.0-1.0)
    float qblur;      ///< amount of qscale smoothing over time (0.0-1.0)

    /**
     * minimum quantizer
     * - encoding: Set by user.
     * - decoding: unused
     */
    int qmin;

    /**
     * maximum quantizer
     * - encoding: Set by user.
     * - decoding: unused
     */
    int qmax;

    /**
     * maximum quantizer difference between frames
     * maximum quantizer difference between frames
     * - encoding: Set by user.
     * - decoding: unused
     */
    int max_qdiff;

    /**
     * ratecontrol qmin qmax limiting method
     * 0-> clipping, 1-> use a nice continuous function to limit qscale wthin qmin/qmax.
     * - encoding: Set by user.
     * - decoding: unused
     */
    float rc_qsquish;

    float rc_qmod_amp;
    int rc_qmod_freq;

    /**
     * decoder bitstream buffer size
     * - encoding: Set by user.
     * - decoding: unused
     */
    int rc_buffer_size;

    /**
     * ratecontrol override, see RcOverride
     * - encoding: Allocated/set/freed by user.
     * - decoding: unused
     */
    int rc_override_count;
    RcOverride *rc_override;

    /**
     * rate control equation
     * - encoding: Set by user
     * - decoding: unused
     */
    const char *rc_eq;

    /**
     * maximum bitrate
     * - encoding: Set by user.
     * - decoding: unused
     */
    int rc_max_rate;

    /**
     * minimum bitrate
     * - encoding: Set by user.
     * - decoding: unused
     */
    int rc_min_rate;

    float rc_buffer_aggressivity;

    /**
     * initial complexity for pass1 ratecontrol
     * - encoding: Set by user.
     * - decoding: unused
     */
    float rc_initial_cplx;

    /**
     * Ratecontrol attempt to use, at maximum, <value> of what can be used without an underflow.
     * - encoding: Set by user.
     * - decoding: unused.
     */
    float rc_max_available_vbv_use;

    /**
     * Ratecontrol attempt to use, at least, <value> times the amount needed to prevent a vbv overflow.
     * - encoding: Set by user.
     * - decoding: unused.
     */
    float rc_min_vbv_overflow_use;

    /**
     * Number of bits which should be loaded into the rc buffer before decoding starts.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int rc_initial_buffer_occupancy;

#define FF_CODER_TYPE_VLC       0
#define FF_CODER_TYPE_AC        1
#define FF_CODER_TYPE_RAW       2
#define FF_CODER_TYPE_RLE       3
#define FF_CODER_TYPE_DEFLATE   4
    /**
     * coder type
     * - encoding: Set by user.
     * - decoding: unused
```

```
      * - decoding: unused
     */
    int coder_type;

    /**
     * context model
     * - encoding: Set by user.
     * - decoding: unused
     */
    int context_model;

    /**
     * minimum Lagrange multipler
     * - encoding: Set by user.
     * - decoding: unused
     */
    int lmin;

    /**
     * maximum Lagrange multipler
     * - encoding: Set by user.
     * - decoding: unused
     */
    int lmax;

    /**
     * frame skip threshold
     * - encoding: Set by user.
     * - decoding: unused
     */
    int frame_skip_threshold;

    /**
     * frame skip factor
     * - encoding: Set by user.
     * - decoding: unused
     */
    int frame_skip_factor;

    /**
     * frame skip exponent
     * - encoding: Set by user.
     * - decoding: unused
     */
    int frame_skip_exp;

    /**
     * frame skip comparison function
     * - encoding: Set by user.
     * - decoding: unused
     */
    int frame_skip_cmp;

    /**
     * trellis RD quantization
     * - encoding: Set by user.
     * - decoding: unused
     */
    int trellis;

    /**
     * - encoding: Set by user.
     * - decoding: unused
     */
    int min_prediction_order;

    /**
     * - encoding: Set by user.
     * - decoding: unused
     */
    int max_prediction_order;

    /**
     * GOP timecode frame start number
     * - encoding: Set by user, in non drop frame format
     * - decoding: Set by libavcodec (timecode in the 25 bits format, -1 if unset)
     */
    int64_t timecode_frame_start;

    /* The RTP callback: This function is called    */
    /* every time the encoder has a packet to send. */
    /* It depends on the encoder if the data starts */
    /* with a Start Code (it should). H.263 does.   */
    /* mb_nb contains the number of macroblocks     */
    /* encoded in the RTP payload.                  */
    void (*rtp_callback)(struct AVCodecContext *avctx, void *data, int size, int mb_nb);

    int rtp_payload_size;   /* The size of the RTP payload: the coder will  */
                            /* do its best to deliver a chunk with size     */
                            /* below rtp_payload_size, the chunk will start */
                            /* with a start code on some codecs like H.263. */
                            /* This doesn't take account of any particular  */
```

```
                             /* headers inside the transmitted RTP payload.  */

    /* statistics, used for 2-pass encoding */
    int mv_bits;
    int header_bits;
    int i_tex_bits;
    int p_tex_bits;
    int i_count;
    int p_count;
    int skip_count;
    int misc_bits;

    /**
     * number of bits used for the previously encoded frame
     * - encoding: Set by libavcodec.
     * - decoding: unused
     */
    int frame_bits;

    /**
     * pass1 encoding statistics output buffer
     * - encoding: Set by libavcodec.
     * - decoding: unused
     */
    char *stats_out;

    /**
     * pass2 encoding statistics input buffer
     * Concatenated stuff from stats_out of pass1 should be placed here.
     * - encoding: Allocated/set/freed by user.
     * - decoding: unused
     */
    char *stats_in;

    /**
     * Work around bugs in encoders which sometimes cannot be detected automatically.
     * - encoding: Set by user
     * - decoding: Set by user
     */
    int workaround_bugs;
#define FF_BUG_AUTODETECT       1  ///< autodetection
#define FF_BUG_OLD_MSMPEG4      2
#define FF_BUG_XVID_ILACE       4
#define FF_BUG_UMP4             8
#define FF_BUG_NO_PADDING       16
#define FF_BUG_AMV              32
#define FF_BUG_AC_VLC           0  ///< Will be removed, libavcodec can now handle these non-compliant files by default.
#define FF_BUG_QPEL_CHROMA      64
#define FF_BUG_STD_QPEL         128
#define FF_BUG_QPEL_CHROMA2     256
#define FF_BUG_DIRECT_BLOCKSIZE 512
#define FF_BUG_EDGE             1024
#define FF_BUG_HPEL_CHROMA      2048
#define FF_BUG_DC_CLIP          4096
#define FF_BUG_MS               8192 ///< Work around various bugs in Microsoft's broken decoders.
#define FF_BUG_TRUNCATED       16384

    /**
     * strictly follow the standard (MPEG4, ...).
     * - encoding: Set by user.
     * - decoding: Set by user.
     * Setting this to STRICT or higher means the encoder and decoder will
     * generally do stupid things, whereas setting it to unofficial or lower
     * will mean the encoder might produce output that is not supported by all
     * spec-compliant decoders. Decoders don't differentiate between normal,
     * unofficial and experimental (that is, they always try to decode things
     * when they can) unless they are explicitly asked to behave stupidly
     * (=strictly conform to the specs)
     */
    int strict_std_compliance;
#define FF_COMPLIANCE_VERY_STRICT   2 ///< Strictly conform to an older more strict version of the spec or reference software.
#define FF_COMPLIANCE_STRICT        1 ///< Strictly conform to all the things in the spec no matter what consequences.
#define FF_COMPLIANCE_NORMAL        0
#define FF_COMPLIANCE_UNOFFICIAL   -1 ///< Allow unofficial extensions
#define FF_COMPLIANCE_EXPERIMENTAL -2 ///< Allow nonstandardized experimental things.

    /**
     * error concealment flags
     * - encoding: unused
     * - decoding: Set by user.
     */
    int error_concealment;
#define FF_EC_GUESS_MVS   1
#define FF_EC_DEBLOCK     2

    /**
     * debug
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int debug;
```

```
#define FF_DEBUG_PICT_INFO   1
#define FF_DEBUG_RC          2
#define FF_DEBUG_BITSTREAM   4
#define FF_DEBUG_MB_TYPE     8
#define FF_DEBUG_QP          16
#define FF_DEBUG_MV          32
#define FF_DEBUG_DCT_COEFF   0x00000040
#define FF_DEBUG_SKIP        0x00000080
#define FF_DEBUG_STARTCODE   0x00000100
#define FF_DEBUG_PTS         0x00000200
#define FF_DEBUG_ER          0x00000400
#define FF_DEBUG_MMCO        0x00000800
#define FF_DEBUG_BUGS        0x00001000
#define FF_DEBUG_VIS_QP      0x00002000
#define FF_DEBUG_VIS_MB_TYPE 0x00004000
#define FF_DEBUG_BUFFERS     0x00008000
#define FF_DEBUG_THREADS     0x00010000

    /**
     * debug
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int debug_mv;
#define FF_DEBUG_VIS_MV_P_FOR  0x00000001 //visualize forward predicted MVs of P frames
#define FF_DEBUG_VIS_MV_B_FOR  0x00000002 //visualize forward predicted MVs of B frames
#define FF_DEBUG_VIS_MV_B_BACK 0x00000004 //visualize backward predicted MVs of B frames

    /**
     * Error recognition; may misdetect some more or less valid parts as errors.
     * - encoding: unused
     * - decoding: Set by user.
     */
    int err_recognition;
#define AV_EF_CRCCHECK  (1<<0)
#define AV_EF_BITSTREAM (1<<1)
#define AV_EF_BUFFER    (1<<2)
#define AV_EF_EXPLODE   (1<<3)

#define AV_EF_CAREFUL    (1<<16)
#define AV_EF_COMPLIANT  (1<<17)
#define AV_EF_AGGRESSIVE (1<<18)


    /**
     * opaque 64bit number (generally a PTS) that will be reordered and
     * output in AVFrame.reordered_opaque
     * @deprecated in favor of pkt_pts
     * - encoding: unused
     * - decoding: Set by user.
     */
    int64_t reordered_opaque;

    /**
     * Hardware accelerator in use
     * - encoding: unused.
     * - decoding: Set by libavcodec
     */
    struct AVHWAccel *hwaccel;

    /**
     * Hardware accelerator context.
     * For some hardware accelerators, a global context needs to be
     * provided by the user. In that case, this holds display-dependent
     * data FFmpeg cannot instantiate itself. Please refer to the
     * FFmpeg HW accelerator documentation to know how to fill this
     * is. e.g. for VA API, this is a struct vaapi_context.
     * - encoding: unused
     * - decoding: Set by user
     */
    void *hwaccel_context;

    /**
     * error
     * - encoding: Set by libavcodec if flags&CODEC_FLAG_PSNR.
     * - decoding: unused
     */
    uint64_t error[AV_NUM_DATA_POINTERS];

    /**
     * DCT algorithm, see FF_DCT_* below
     * - encoding: Set by user.
     * - decoding: unused
     */
    int dct_algo;
#define FF_DCT_AUTO    0
#define FF_DCT_FASTINT 1
#define FF_DCT_INT     2
#define FF_DCT_MMX     3
#define FF_DCT_ALTIVEC 5
#define FF_DCT_FAAN    6
```

```c

    /**
     * IDCT algorithm, see FF_IDCT_* below.
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int idct_algo;
#define FF_IDCT_AUTO          0
#define FF_IDCT_INT           1
#define FF_IDCT_SIMPLE        2
#define FF_IDCT_SIMPLEMMX     3
#define FF_IDCT_LIBMPEG2MMX   4
#define FF_IDCT_MMI           5
#define FF_IDCT_ARM           7
#define FF_IDCT_ALTIVEC       8
#define FF_IDCT_SH4           9
#define FF_IDCT_SIMPLEARM     10
#define FF_IDCT_H264          11
#define FF_IDCT_VP3           12
#define FF_IDCT_IPP           13
#define FF_IDCT_XVIDMMX       14
#define FF_IDCT_CAVS          15
#define FF_IDCT_SIMPLEARMV5TE 16
#define FF_IDCT_SIMPLEARMV6   17
#define FF_IDCT_SIMPLEVIS     18
#define FF_IDCT_WMV2          19
#define FF_IDCT_FAAN          20
#define FF_IDCT_EA            21
#define FF_IDCT_SIMPLENEON    22
#define FF_IDCT_SIMPLEALPHA   23
#define FF_IDCT_BINK          24

#if FF_API_DSP_MASK
    /**
     * Unused.
     * @deprecated use av_set_cpu_flags_mask() instead.
     */
    attribute_deprecated unsigned dsp_mask;
#endif

    /**
     * bits per sample/pixel from the demuxer (needed for huffyuv).
     * - encoding: Set by libavcodec.
     * - decoding: Set by user.
     */
     int bits_per_coded_sample;

    /**
     * Bits per sample/pixel of internal libavcodec pixel/sample format.
     * - encoding: set by user.
     * - decoding: set by libavcodec.
     */
    int bits_per_raw_sample;

    /**
     * low resolution decoding, 1-> 1/2 size, 2->1/4 size
     * - encoding: unused
     * - decoding: Set by user.
     */
     int lowres;

    /**
     * the picture in the bitstream
     * - encoding: Set by libavcodec.
     * - decoding: Set by libavcodec.
     */
    AVFrame *coded_frame;

    /**
     * thread count
     * is used to decide how many independent tasks should be passed to execute()
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int thread_count;

    /**
     * Which multithreading methods to use.
     * Use of FF_THREAD_FRAME will increase decoding delay by one frame per thread,
     * so clients which cannot provide future frames should not use it.
     *
     * - encoding: Set by user, otherwise the default is used.
     * - decoding: Set by user, otherwise the default is used.
     */
    int thread_type;
#define FF_THREAD_FRAME   1 ///< Decode more than one frame at once
#define FF_THREAD_SLICE   2 ///< Decode more than one part of a single frame at once

    /**
     * Which multithreading methods are in use by the codec.
     * - encoding: Set by libavcodec.
     * - decoding: Set by libavcodec.
     */
    int active_thread_type;

    /**
     * Set by the client if its custom get_buffer() callback can be called
     * synchronously from another thread, which allows faster multithreaded decoding.
     * draw_horiz_band() will be called from other threads regardless of this setting.
     * Ignored if the default get_buffer() is used.
     * - encoding: Set by user.
     * - decoding: Set by user.
     */
    int thread_safe_callbacks;

    /**
     * The codec may call this to execute several independent things.
     * It will return only after finishing all tasks.
     * The user may replace this with some multithreaded implementation,
     * the default implementation will execute the parts serially.
     * @param count the number of things to execute
     * - encoding: Set by libavcodec, user can override.
     * - decoding: Set by libavcodec, user can override.
     */
    int (*execute)(struct AVCodecContext *c, int (*func)(struct AVCodecContext *c2, void *arg), void *arg2, int *ret, int count, int size);

    /**
     * The codec may call this to execute several independent things.
     * It will return only after finishing all tasks.
     * The user may replace this with some multithreaded implementation,
     * the default implementation will execute the parts serially.
     * Also see avcodec_thread_init and e.g. the --enable-pthread configure option.
     * @param c context passed also to func
     * @param count the number of things to execute
     * @param arg2 argument passed unchanged to func
     * @param ret return values of executed functions, must have space for "count" values. May be NULL.
     * @param func function that will be called count times, with jobnr from 0 to count-1.
     *             threadnr will be in the range 0 to c->thread_count-1 < MAX_THREADS and so that no
     *             two instances of func executing at the same time will have the same threadnr.
     * @return always 0 currently, but code should handle a future improvement where when any call to func
     *         returns < 0 no further calls to func may be done and < 0 is returned.
     * - encoding: Set by libavcodec, user can override.
     * - decoding: Set by libavcodec, user can override.
     */
    int (*execute2)(struct AVCodecContext *c, int (*func)(struct AVCodecContext *c2, void *arg, int jobnr, int threadnr), void *arg2, int *ret, int count);

    /**
     * thread opaque
     * Can be used by execute() to store some per AVCodecContext stuff.
     * - encoding: set by execute()
     * - decoding: set by execute()
     */
    void *thread_opaque;

    /**
     * noise vs. sse weight for the nsse comparsion function
     * - encoding: Set by user.
     * - decoding: unused
     */
     int nsse_weight;

    /**
     * profile
     * - encoding: Set by user.
     * - decoding: Set by libavcodec.
     */
     int profile;
#define FF_PROFILE_UNKNOWN -99
#define FF_PROFILE_RESERVED -100

#define FF_PROFILE_AAC_MAIN 0
#define FF_PROFILE_AAC_LOW  1
#define FF_PROFILE_AAC_SSR  2
#define FF_PROFILE_AAC_LTP  3
#define FF_PROFILE_AAC_HE   4
#define FF_PROFILE_AAC_HE_V2 28
#define FF_PROFILE_AAC_LD   22
#define FF_PROFILE_AAC_ELD  38

#define FF_PROFILE_DTS         20
#define FF_PROFILE_DTS_ES      30
#define FF_PROFILE_DTS_96_24   40
#define FF_PROFILE_DTS_HD_HRA  50
#define FF_PROFILE_DTS_HD_MA   60

#define FF_PROFILE_MPEG2_422    0
#define FF_PROFILE_MPEG2_HIGH   1
#define FF_PROFILE_MPEG2_SS     2
#define FF_PROFILE_MPEG2_SNR_SCALABLE  3
#define FF_PROFILE_MPEG2_MAIN   4
#define FF_PROFILE_MPEG2_SIMPLE 5
```

```
#define FF_PROFILE_MPEG2_SIMPLE 5

#define FF_PROFILE_H264_CONSTRAINED  (1<<9)  // 8+1; constraint_set1_flag
#define FF_PROFILE_H264_INTRA        (1<<11) // 8+3; constraint_set3_flag

#define FF_PROFILE_H264_BASELINE             66
#define FF_PROFILE_H264_CONSTRAINED_BASELINE (66|FF_PROFILE_H264_CONSTRAINED)
#define FF_PROFILE_H264_MAIN                 77
#define FF_PROFILE_H264_EXTENDED             88
#define FF_PROFILE_H264_HIGH                 100
#define FF_PROFILE_H264_HIGH_10              110
#define FF_PROFILE_H264_HIGH_10_INTRA        (110|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_HIGH_422             122
#define FF_PROFILE_H264_HIGH_422_INTRA       (122|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_HIGH_444             144
#define FF_PROFILE_H264_HIGH_444_PREDICTIVE  244
#define FF_PROFILE_H264_HIGH_444_INTRA       (244|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_CAVLC_444            44

#define FF_PROFILE_VC1_SIMPLE   0
#define FF_PROFILE_VC1_MAIN     1
#define FF_PROFILE_VC1_COMPLEX  2
#define FF_PROFILE_VC1_ADVANCED 3

#define FF_PROFILE_MPEG4_SIMPLE                     0
#define FF_PROFILE_MPEG4_SIMPLE_SCALABLE            1
#define FF_PROFILE_MPEG4_CORE                       2
#define FF_PROFILE_MPEG4_MAIN                       3
#define FF_PROFILE_MPEG4_N_BIT                      4
#define FF_PROFILE_MPEG4_SCALABLE_TEXTURE           5
#define FF_PROFILE_MPEG4_SIMPLE_FACE_ANIMATION      6
#define FF_PROFILE_MPEG4_BASIC_ANIMATED_TEXTURE     7
#define FF_PROFILE_MPEG4_HYBRID                     8
#define FF_PROFILE_MPEG4_ADVANCED_REAL_TIME         9
#define FF_PROFILE_MPEG4_CORE_SCALABLE             10
#define FF_PROFILE_MPEG4_ADVANCED_CODING           11
#define FF_PROFILE_MPEG4_ADVANCED_CORE             12
#define FF_PROFILE_MPEG4_ADVANCED_SCALABLE_TEXTURE 13
#define FF_PROFILE_MPEG4_SIMPLE_STUDIO             14
#define FF_PROFILE_MPEG4_ADVANCED_SIMPLE           15

    /**
     * level
     * - encoding: Set by user.
     * - decoding: Set by libavcodec.
     */
     int level;
#define FF_LEVEL_UNKNOWN -99

    /**
     *
     * - encoding: unused
     * - decoding: Set by user.
     */
    enum AVDiscard skip_loop_filter;

    /**
     *
     * - encoding: unused
     * - decoding: Set by user.
     */
    enum AVDiscard skip_idct;

    /**
     *
     * - encoding: unused
     * - decoding: Set by user.
     */
    enum AVDiscard skip_frame;

    /**
     * Header containing style information for text subtitles.
     * For SUBTITLE_ASS subtitle type, it should contain the whole ASS
     * [Script Info] and [V4+ Styles] section, plus the [Events] line and
     * the Format line following. It shouldn't include any Dialogue line.
     * - encoding: Set/allocated/freed by user (before avcodec_open2())
     * - decoding: Set/allocated/freed by libavcodec (by avcodec_open2())
     */
    uint8_t *subtitle_header;
    int subtitle_header_size;

    /**
     * Simulates errors in the bitstream to test error concealment.
     * - encoding: Set by user.
     * - decoding: unused
     */
    int error_rate;

    /**
     * Current packet as passed into the decoder, to avoid having
     * to pass the packet into every function. Currently only valid
     * inside lavc and get/release buffer callbacks.
     * inside lavc and get/release_buffer callbacks.
     * - decoding: set by avcodec_decode_*, read by get_buffer() for setting pkt_pts
     * - encoding: unused
     */
    AVPacket *pkt;

    /**
     * VBV delay coded in the last frame (in periods of a 27 MHz clock).
     * Used for compliant TS muxing.
     * - encoding: Set by libavcodec.
     * - decoding: unused.
     */
    uint64_t vbv_delay;

    /**
     * Timebase in which pkt_dts/pts and AVPacket.dts/pts are.
     * Code outside libavcodec should access this field using:
     * avcodec_set_pkt_timebase(avctx)
     * - encoding unused.
     * - decodimg set by user
     */
    AVRational pkt_timebase;

    /**
     * AVCodecDescriptor
     * Code outside libavcodec should access this field using:
     * avcodec_get_codec_descriptor(avctx)
     * - encoding: unused.
     * - decoding: set by libavcodec.
     */
    const AVCodecDescriptor *codec_descriptor;

    /**
     * Current statistics for PTS correction.
     * - decoding: maintained and used by libavcodec, not intended to be used by user apps
     * - encoding: unused
     */
    int64_t pts_correction_num_faulty_pts; /// Number of incorrect PTS values so far
    int64_t pts_correction_num_faulty_dts; /// Number of incorrect DTS values so far
    int64_t pts_correction_last_pts;       /// PTS of the last frame
    int64_t pts_correction_last_dts;       /// DTS of the last frame
} AVCodecContext;
```

光定义就真是够多的。下面挑一些关键的变量来看看（这里只考虑解码）。

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）

在这里需要注意：AVCodecContext中很多的参数是编码的时候使用的，而不是解码的时候使用的。

其实这些参数都比较容易理解。就不多费篇幅了。在这里看一下以下几个参数：

1.codec_type

编解码器类型有以下几种：

[cpp]

```cpp
enum AVMediaType {
    AVMEDIA_TYPE_UNKNOWN = -1,  ///< Usually treated as AVMEDIA_TYPE_DATA
    AVMEDIA_TYPE_VIDEO,
    AVMEDIA_TYPE_AUDIO,
    AVMEDIA_TYPE_DATA,          ///< Opaque data information usually continuous
    AVMEDIA_TYPE_SUBTITLE,
    AVMEDIA_TYPE_ATTACHMENT,    ///< Opaque data information usually sparse
    AVMEDIA_TYPE_NB
};
```

2.sample_fmt

在FFMPEG中音频采样格式有以下几种：

[cpp]

```cpp
enum AVSampleFormat {
    AV_SAMPLE_FMT_NONE = -1,
    AV_SAMPLE_FMT_U8,          ///< unsigned 8 bits
    AV_SAMPLE_FMT_S16,         ///< signed 16 bits
    AV_SAMPLE_FMT_S32,         ///< signed 32 bits
    AV_SAMPLE_FMT_FLT,         ///< float
    AV_SAMPLE_FMT_DBL,         ///< double

    AV_SAMPLE_FMT_U8P,         ///< unsigned 8 bits, planar
    AV_SAMPLE_FMT_S16P,        ///< signed 16 bits, planar
    AV_SAMPLE_FMT_S32P,        ///< signed 32 bits, planar
    AV_SAMPLE_FMT_FLTP,        ///< float, planar
    AV_SAMPLE_FMT_DBLP,        ///< double, planar

    AV_SAMPLE_FMT_NB           ///< Number of sample formats. DO NOT USE if linking dynamically
};
```

3.profile

在FFMPEG中型有以下几种，可以看出AAC，MPEG2，H.264，VC-1，MPEG4都有型的概念。

[cpp]

```cpp
#define FF_PROFILE_UNKNOWN -99
#define FF_PROFILE_RESERVED -100

#define FF_PROFILE_AAC_MAIN 0
#define FF_PROFILE_AAC_LOW  1
#define FF_PROFILE_AAC_SSR  2
#define FF_PROFILE_AAC_LTP  3
#define FF_PROFILE_AAC_HE   4
#define FF_PROFILE_AAC_HE_V2 28
#define FF_PROFILE_AAC_LD   22
#define FF_PROFILE_AAC_ELD  38

#define FF_PROFILE_DTS         20
#define FF_PROFILE_DTS_ES      30
#define FF_PROFILE_DTS_96_24   40
#define FF_PROFILE_DTS_HD_HRA  50
#define FF_PROFILE_DTS_HD_MA   60

#define FF_PROFILE_MPEG2_422    0
#define FF_PROFILE_MPEG2_HIGH   1
#define FF_PROFILE_MPEG2_SS     2
#define FF_PROFILE_MPEG2_SNR_SCALABLE  3
#define FF_PROFILE_MPEG2_MAIN   4
#define FF_PROFILE_MPEG2_SIMPLE 5

#define FF_PROFILE_H264_CONSTRAINED  (1<<9)  // 8+1; constraint_set1_flag
#define FF_PROFILE_H264_INTRA        (1<<11) // 8+3; constraint_set3_flag

#define FF_PROFILE_H264_BASELINE             66
#define FF_PROFILE_H264_CONSTRAINED_BASELINE (66|FF_PROFILE_H264_CONSTRAINED)
#define FF_PROFILE_H264_MAIN                 77
#define FF_PROFILE_H264_EXTENDED             88
#define FF_PROFILE_H264_HIGH                 100
#define FF_PROFILE_H264_HIGH_10              110
#define FF_PROFILE_H264_HIGH_10_INTRA        (110|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_HIGH_422             122
#define FF_PROFILE_H264_HIGH_422_INTRA       (122|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_HIGH_444             144
#define FF_PROFILE_H264_HIGH_444_PREDICTIVE  244
#define FF_PROFILE_H264_HIGH_444_INTRA       (244|FF_PROFILE_H264_INTRA)
#define FF_PROFILE_H264_CAVLC_444            44

#define FF_PROFILE_VC1_SIMPLE   0
#define FF_PROFILE_VC1_MAIN     1
#define FF_PROFILE_VC1_COMPLEX  2
#define FF_PROFILE_VC1_ADVANCED 3

#define FF_PROFILE_MPEG4_SIMPLE                     0
#define FF_PROFILE_MPEG4_SIMPLE_SCALABLE            1
#define FF_PROFILE_MPEG4_CORE                       2
#define FF_PROFILE_MPEG4_MAIN                       3
#define FF_PROFILE_MPEG4_N_BIT                      4
#define FF_PROFILE_MPEG4_SCALABLE_TEXTURE           5
#define FF_PROFILE_MPEG4_SIMPLE_FACE_ANIMATION      6
#define FF_PROFILE_MPEG4_BASIC_ANIMATED_TEXTURE     7
#define FF_PROFILE_MPEG4_HYBRID                     8
#define FF_PROFILE_MPEG4_ADVANCED_REAL_TIME         9
#define FF_PROFILE_MPEG4_CORE_SCALABLE             10
#define FF_PROFILE_MPEG4_ADVANCED_CODING           11
#define FF_PROFILE_MPEG4_ADVANCED_CORE             12
#define FF_PROFILE_MPEG4_ADVANCED_SCALABLE_TEXTURE 13
#define FF_PROFILE_MPEG4_SIMPLE_STUDIO             14
#define FF_PROFILE_MPEG4_ADVANCED_SIMPLE           15
```

文章标签： ffmpeg AVCodecContext 源代码 解码 视频

个人分类： FFMPEG

所属专栏： FFmpeg