原 FFmpeg源代码简单分析：av_write_trailer()

2015年03月11日 17:29:06　阅读数：14963

==========

FFmpeg的库函数源代码分析文章列表：

【架构图】

FFmpeg 源代码结构图 - 解码

FFmpeg 源代码结构图 - 编码

【通用】

FFmpeg 源代码简单分析： av_register_all()

FFmpeg 源代码简单分析： avcodec_register_all()

FFmpeg 源代码简单分析：内存的分配和释放（ av_malloc()、 av_free() 等）

FFmpeg 源代码简单分析：常见结构体的初始化和销毁（ AVFormatContext， AVFrame 等）

FFmpeg 源代码简单分析： avio_open2()

FFmpeg 源代码简单分析： av_find_decoder() 和 av_find_encoder()

FFmpeg 源代码简单分析： avcodec_open2()

FFmpeg 源代码简单分析： avcodec_close()

【解码】

图解 FFMPEG 打开媒体的函数 avformat_open_input

FFmpeg 源代码简单分析： avformat_open_input()

FFmpeg 源代码简单分析： avformat_find_stream_info()

FFmpeg 源代码简单分析： av_read_frame()

FFmpeg 源代码简单分析： avcodec_decode_video2()

FFmpeg 源代码简单分析： avformat_close_input()

【编码】

FFmpeg 源代码简单分析： avformat_alloc_output_context2()

FFmpeg 源代码简单分析： avformat_write_header()

FFmpeg 源代码简单分析： avcodec_encode_video()

FFmpeg 源代码简单分析： av_write_frame()

FFmpeg 源代码简单分析： av_write_trailer()

【其它】

FFmpeg 源代码简单分析：日志输出系统（ av_log() 等）

FFmpeg 源代码简单分析：结构体成员管理系统 -AVClass

FFmpeg 源代码简单分析：结构体成员管理系统 -AVOption

FFmpeg 源代码简单分析： libswscale 的 sws_getContext()

FFmpeg 源代码简单分析： libswscale 的 sws_scale()

FFmpeg 源代码简单分析： libavdevice 的 avdevice_register_all()

FFmpeg 源代码简单分析： libavdevice 的 gdigrab

【脚本】

FFmpeg 源代码简单分析： makefile

FFmpeg 源代码简单分析： configure

【H.264】

FFmpeg 的 H.264 解码器源代码简单分析：概述

==========

打算写两篇文章简单分析FFmpeg的写文件用到的3个函数avformat_write_header()，av_write_frame()以及av_write_trailer()。这篇文章继续分析av_write_trailer()。

av_write_trailer()用于输出文件尾，它的声明位于libavformat\avformat.h，如下所示。

```cpp
/**
 * Write the stream trailer to an output media file and free the
 * file private data.
 *
 * May only be called after a successful call to avformat_write_header.
 *
 * @param s media file handle
 * @return 0 if OK, AVERROR_xxx on error
 */
int av_write_trailer(AVFormatContext *s);
```

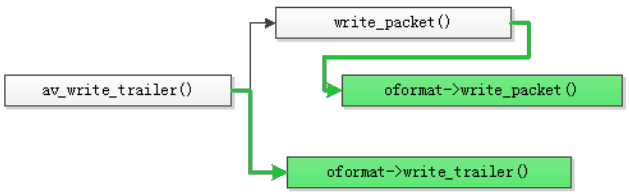
它只需要指定一个参数，即用于输出的AVFormatContext。
函数正常执行后返回值等于0。

这2个函数最典型的例子可以参考：

最简单的基于FFMPEG的视频编码器（YUV编码为H.264）

函数调用关系图

av_write_trailer()的调用关系如下图所示。



struct AVOutputFormat *oformat;

雷霄骅 (Lei Xiaohua)
leixiaohua1020@126.com
http://blog.csdn.net/leixiaohua1020

av_write_trailer()

av_write_trailer()的定义位于libavformat\mux.c，如下所示。

```cpp
int av_write_trailer(AVFormatContext *s)
{
    int ret, i;

    for (;; ) {
        AVPacket pkt;
        ret = interleave_packet(s, &pkt, NULL, 1);
        if (ret < 0)
            goto fail;
        if (!ret)
            break;
        //写入AVPacket
        ret = write_packet(s, &pkt);
        if (ret >= 0)
            s->streams[pkt.stream_index]->nb_frames++;

        av_free_packet(&pkt);

        if (ret < 0)
            goto fail;
        if(s->pb && s->pb->error)
            goto fail;
    }

fail:
    //写文件尾
    if (s->oformat->write_trailer)
        if (ret >= 0) {
        ret = s->oformat->write_trailer(s);
        } else {
            s->oformat->write_trailer(s);
        }

    if (s->pb)
       avio_flush(s->pb);
    if (ret == 0)
       ret = s->pb ? s->pb->error : 0;
    for (i = 0; i < s->nb_streams; i++) {
        av_freep(&s->streams[i]->priv_data);
        av_freep(&s->streams[i]->index_entries);
    }
    if (s->oformat->priv_class)
        av_opt_free(s->priv_data);
    av_freep(&s->priv_data);
    return ret;
}
```

从源代码可以看出av_write_trailer()主要完成了以下两步工作：

（1）循环调用interleave_packet()以及write_packet()，将还未输出的AVPacket输出出来。

（2）调用AVOutputFormat的write_trailer()，输出文件尾。

其中第一步和av_write_frame()中的步骤大致是一样的（interleave_packet()这一部分在并不包含在av_write_frame()中，而是包含在av_interleaved_write_frame()中，这一部分源代码还没有分析），可以参考文章《 FFmpeg源代码简单分析：av_write_frame() 》。下面分析一下第二步。

AVOutputFormat->write_trailer()

AVOutputFormat的write_trailer()是一个函数指针，指向特定的AVOutputFormat中的实现函数。我们以FLV对应的AVOutputFormat为例，看一下它的定义，如下所示。

```cpp
AVOutputFormat ff_flv_muxer = {
    .name           = "flv",
    .long_name      = NULL_IF_CONFIG_SMALL("FLV (Flash Video)"),
    .mime_type      = "video/x-flv",
    .extensions     = "flv",
    .priv_data_size = sizeof(FLVContext),
    .audio_codec    = CONFIG_LIBMP3LAME ? AV_CODEC_ID_MP3 : AV_CODEC_ID_ADPCM_SWF,
    .video_codec    = AV_CODEC_ID_FLV1,
    .write_header   = flv_write_header,
    .write_packet   = flv_write_packet,
    .write_trailer  = flv_write_trailer,
    .codec_tag      = (const AVCodecTag* const []) {
                          flv_video_codec_ids, flv_audio_codec_ids, 0
                      },
    .flags          = AVFMT_GLOBALHEADER | AVFMT_VARIABLE_FPS |
                      AVFMT_TS_NONSTRICT,
};
```

从FLV对应的AVOutputFormat结构体的定义我们可以看出，write_trailer()指向了flv_write_trailer()函数。

flv_write_trailer()

flv_write_trailer()函数的定义位于libavformat\flvenc.c，如下所示。

```cpp
static int flv_write_trailer(AVFormatContext *s)
{
    int64_t file_size;

    AVIOContext *pb = s->pb;
    FLVContext *flv = s->priv_data;
    int i;

    /* Add EOS tag */
    for (i = 0; i < s->nb_streams; i++) {
        AVCodecContext *enc = s->streams[i]->codec;
        FLVStreamContext *sc = s->streams[i]->priv_data;
        if (enc->codec_type == AVMEDIA_TYPE_VIDEO &&
                (enc->codec_id == AV_CODEC_ID_H264 || enc->codec_id == AV_CODEC_ID_MPEG4))
            put_avc_eos_tag(pb, sc->last_ts);
    }

    file_size = avio_tell(pb);

    /* update information */
    if (avio_seek(pb, flv->duration_offset, SEEK_SET) < 0)
        av_log(s, AV_LOG_WARNING, "Failed to update header with correct duration.\n");
    else
        put_amf_double(pb, flv->duration / (double)1000);
    if (avio_seek(pb, flv->filesize_offset, SEEK_SET) < 0)
        av_log(s, AV_LOG_WARNING, "Failed to update header with correct filesize.\n");
    else
        put_amf_double(pb, file_size);

    avio_seek(pb, file_size, SEEK_SET);
    return 0;
}
```

从flv_write_trailer()的源代码可以看出该函数做了以下两步工作：
（1）如果视频流是H.264，则添加包含EOS（End Of Stream） NALU的Tag。
（2）更新FLV的时长信息，以及文件大小信息。
其中，put_avc_eos_tag()函数用于添加包含EOS NALU的Tag（包含结尾的一个PreviousTagSize），如下所示。

```cpp
static void put_avc_eos_tag(AVIOContext *pb, unsigned ts)
{
    avio_w8(pb, FLV_TAG_TYPE_VIDEO);
    avio_wb24(pb, 5);               /* Tag Data Size */
    avio_wb24(pb, ts);              /* lower 24 bits of timestamp in ms */
    avio_w8(pb, (ts >> 24) & 0x7F); /* MSB of ts in ms */
    avio_wb24(pb, 0);               /* StreamId = 0 */
    avio_w8(pb, 23);                /* ub[4] FrameType = 1, ub[4] CodecId = 7 */
    avio_w8(pb, 2);                 /* AVC end of sequence */
    avio_wb24(pb, 0);               /* Always 0 for AVC EOS. */
    avio_wb32(pb, 16);              /* Size of FLV tag */
}
```

可以参考FLV封装格式理解上述函数。由于前面的文章中已经描述过FLV封装格式，在这里不再重复叙述，在这里仅在此记录一下AVCVIDEOPACKET的格式，如下所示。

AVCVIDEOPACKET		
Field	Type	Comment
AVCPacketType	UI8	0: AVC sequence header 1: AVC NALU 2: AVC end of sequence (lower level NALU sequence ender is not required or supported)
CompositionTime	SI24	if AVCPacketType == 1 Composition time offset else 0
Data	UI8[n]	if AVCPacketType == 0 AVCDecoderConfigurationRecord else if AVCPacketType == 1 One or more NALUs (can be individual slices per FLV packets; that is, full frames are not strictly required) else if AVCPacketType == 2 Empty

可以看出包含EOS　NALU的AVCVIDEOPACKET的AVCPacketType为2。在这种情况下，AVCVIDEOPACKET的CompositionTime字段取0，并且

无需包含Data字段。

雷霄骅
leixiaohua1020@126.com
http://blog.csdn.net/leixiaohua1020

文章标签： ffmpeg AVPacket FLV 输出 源代码

个人分类： FFMPEG

所属专栏： FFmpeg