原 FFmpeg源代码结构图 - 编码

2015年03月13日 14:42:15　阅读数：24290

==========

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】

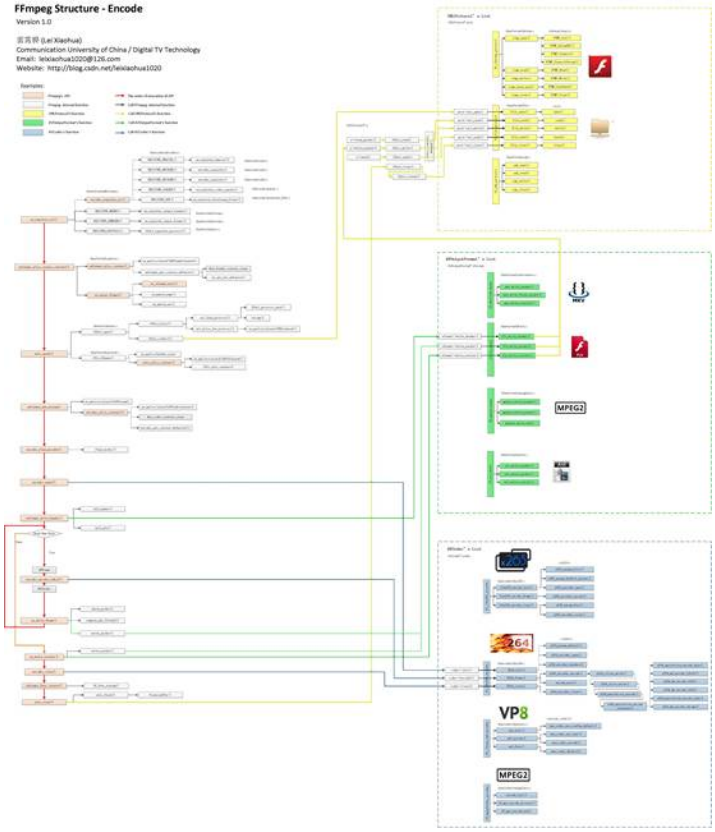
FFmpeg 的 H.264 解码器源代码简单分析：概述

==========

上一篇文章深入分析了FFmpeg解码过程中核心API的内部源代码，本文继续分析FFmpeg编码过程中核心API的内部源代码。本文的编码流程可以参考程序《 最简单的基于FFmpeg的视频编码器 》。

函数调用关系图

首先呈现分析的结果，如图所示。这张图的尺寸很大（大于4000x4000），因此需要打开图片链接之后将图片保存为本地文件，然后才能查看。它表明了FFmpeg在编码一个视频的时候的函数调用流程。为了保证结构清晰，其中仅列出了最关键的函数，剔除了其它不是特别重要的函数。



单击查看更清晰的图片

下面解释一下图中关键标记的含义。

函数背景色

函数在图中以方框的形式表现出来。不同的背景色标志了该函数不同的作用：

粉红色背景函数：FFmpeg的API函数。
白色背景的函数：FFmpeg的内部函数。
黄色背景的函数：URLProtocol结构体中的函数，包含了读写各种协议的功能。
绿色背景的函数：AVOutputFormat结构体中的函数，包含了读写各种封装格式的功能。
蓝色背景的函数：AVCodec结构体中的函数，包含了编解码的功能。

区域

整个关系图可以分为以下几个区域：

左边区域——架构函数区域：这些函数并不针对某一特定的视频格式。
右上方黄色区域——协议处理函数区域：不同的协议（RTP，RTMP，FILE）会调用不同的协议处理函数。
右边中间绿色区域——封装格式处理函数区域：不同的封装格式（MKV，FLV，MPEG2TS，AVI）会调用不同的封装格式处理函数。
右边下方蓝色区域——编解码函数区域：不同的编码标准（HEVC，H.264，MPEG2）会调用不同的编解码函数。

箭头线

为了把调用关系表示的更明显，图中的箭头线也使用了不同的颜色：
红色的箭头线：标志了编码的流程。
其他颜色的箭头线：标志了函数之间的调用关系。其中：
调用URLProtocol结构体中的函数用黄色箭头线标识；
调用AVOutputFormat结构体中的函数用绿色箭头线标识；
调用AVCodec结构体中的函数用蓝色箭头线标识。

函数所在的文件

每个函数标识了它所在的文件路径。

函数功能简述

下面简单列出几个区域中函数之间的调用关系（函数之间的调用关系使用缩进的方式表现出来）。详细的函数分析可以参考相关的《FFmpeg源代码分析》系列文章。

左边区域（架构函数）

1. av_register_all()【函数简单分析】

1) avcodec_register_all()

(a) REGISTER_HWACCEL()

(b) REGISTER_ENCODER()

(c) REGISTER_DECODER()

(d) REGISTER_PARSER()

(e) REGISTER_BSF()

2) REGISTER_MUXER()

3) REGISTER_DEMUXER()

4) REGISTER_PROTOCOL()

2. avformat_alloc_output_context2()【函数简单分析】

1) avformat_alloc_context()

(a) av_malloc(sizeof(AVFormatContext))

(b) avformat_get_context_defaults()

a) av_opt_set_defaults()

2) av_guess_format()

(a) av_oformat_next()

(b) av_match_name()

(c) av_match_ext()

3. avio_open2()【函数简单分析】

1) ffurl_open()

(a) ffurl_alloc()

a) url_find_protocol()

b) url_alloc_for_protocol()

(b) ffurl_connect()

a) URLProtocol->url_open()

2) ffio_fdopen()

(a) av_malloc(buffer_size)

(b) avio_alloc_context()

a) av_mallocz(sizeof(AVIOContext))

b) ffio_init_context()

4. avformat_new_stream()【函数简单分析】

1) av_mallocz(sizeof(AVStream))

2) avcodec_alloc_context3()

(a) av_malloc(sizeof(AVCodecContext))

(b) avcodec_get_context_defaults3()

5. avcodec_find_encoder()【函数简单分析】

1) find_encdec()

6. avcodec_open2()【函数简单分析】

1) AVCodec->init()

7. avformat_write_header()【函数简单分析】

1) init_muxer()

2) AVOutputFormat->write_header()

3) init_pts()

8. avcodec_encode_video2()【函数简单分析】

1) AVCodec->encode2()

9. av_write_frame()【函数简单分析】

1) check_packet()

2) compute_pkt_fields2()

3) write_packet()

(a) AVOutputFormat->write_packet()

10. av_write_trailer()【函数简单分析】

1) write_packet()

2) AVOutputFormat->write_trailer()

11. avcodec_close()【函数简单分析】

1) AVCodec->close()

12. avformat_free_context()【函数简单分析】

1) ff_free_stream()

13. avio_close()【函数简单分析】

1) avio_flush()

(a) flush_buffer()

2) ffurl_close()

(a) ffurl_closep()

a) URLProtocol->url_close()

右上区域（URLProtocol协议处理函数）

URLProtocol结构体包含如下协议处理函数指针：

url_open()：打开
url_read()：读取
url_write()：写入
url_seek()：调整进度

url_close()：关闭

【例子】不同的协议对应着上述接口有不同的实现函数，举几个例子：

File协议（即文件）对应的URLProtocol结构体ff_file_protocol：
url_open() -> file_open() -> open()
url_read() -> file_read() -> read()
url_write() -> file_write() -> write()
url_seek() -> file_seek() -> lseek()
url_close() -> file_close() -> close()

RTMP协议（libRTMP）对应的URLProtocol结构体ff_librtmp_protocol：
url_open() -> rtmp_open() -> RTMP_Init(), RTMP_SetupURL(), RTMP_Connect(), RTMP_ConnectStream()
url_read() -> rtmp_read() -> RTMP_Read()
url_write() -> rtmp_write() -> RTMP_Write()
url_seek() -> rtmp_read_seek() -> RTMP_SendSeek()
url_close() -> rtmp_close() -> RTMP_Close()

UDP协议对应的URLProtocol结构体ff_udp_protocol：
url_open() -> udp_open()
url_read() -> udp_read()
url_write() -> udp_write()
url_seek() -> udp_close()
url_close() -> udp_close()

右中区域（AVOutputFormat封装格式处理函数）

AVOutputFormat包含如下封装格式处理函数指针：

write_header()：写文件头
write_packet()：写一帧数据
write_trailer()：写文件尾

【例子】不同的封装格式对应着上述接口有不同的实现函数，举几个例子：

FLV封装格式对应的AVOutputFormat结构体ff_flv_muxer：
write_header() -> flv_write_header()
write_packet() –> flv_write_packet()
write_trailer() -> flv_write_trailer()

MKV封装格式对应的AVOutputFormat结构体ff_matroska_muxer：
write_header() -> mkv_write_header()
write_packet() –> mkv_write_flush_packet()
write_trailer() -> mkv_write_trailer()

MPEG2TS封装格式对应的AVOutputFormat结构体ff_mpegts_muxer：
write_header() -> mpegts_write_header()
write_packet() –> mpegts_write_packet()
write_trailer() -> mpegts_write_end()

AVI封装格式对应的AVOutputFormat结构体ff_avi_muxer：
write_header() -> avi_write_header()
write_packet() –> avi_write_packet()
write_trailer() -> avi_write_trailer()

右下区域（AVCodec编解码函数）

AVCodec包含如下编解码函数指针：

init()：初始化
encode2()：编码一帧数据
close()：关闭

【例子】不同的编解码器对应着上述接口有不同的实现函数，举几个例子：

HEVC编码器对应的AVCodec结构体ff_libx265_encoder：
init() -> libx265_encode_init() -> x265_param_alloc(), x265_param_default_preset(), x265_encoder_open()
encode2() -> libx265_encode_frame() -> x265_encoder_encode()
close() -> libx265_encode_close() -> x265_param_free(), x265_encoder_close()

H.264编码器对应的AVCodec结构体ff_libx264_encoder：
init() -> X264_init() -> x264_param_default(), x264_encoder_open(), x264_encoder_headers()
encode2() -> X264_frame() -> x264_encoder_encode()
close() -> X264_close() -> x264_encoder_close()

VP8编码器（libVPX）对应的AVCodec结构体ff_libvpx_vp8_encoder：
init() -> vpx_init() -> vpx_codec_enc_config_default()
encode2() -> vp8_encode() -> vpx_codec_enc_init(), vpx_codec_encode()
close() -> vp8_free() -> vpx_codec_destroy()

MPEG2编码器对应的AVCodec结构体ff_mpeg2video_encoder：
init() -> encode_init()
encode2() -> ff_mpv_encode_picture()
close() -> ff_mpv_encode_end()

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

文章标签： FFmpeg 函数调用 源代码 函数
个人分类： FFMPEG
所属专栏： FFmpeg