FFmpeg源代码结构图 - 解码

2015年03月12日 14:03:16　阅读数：58406

==========

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

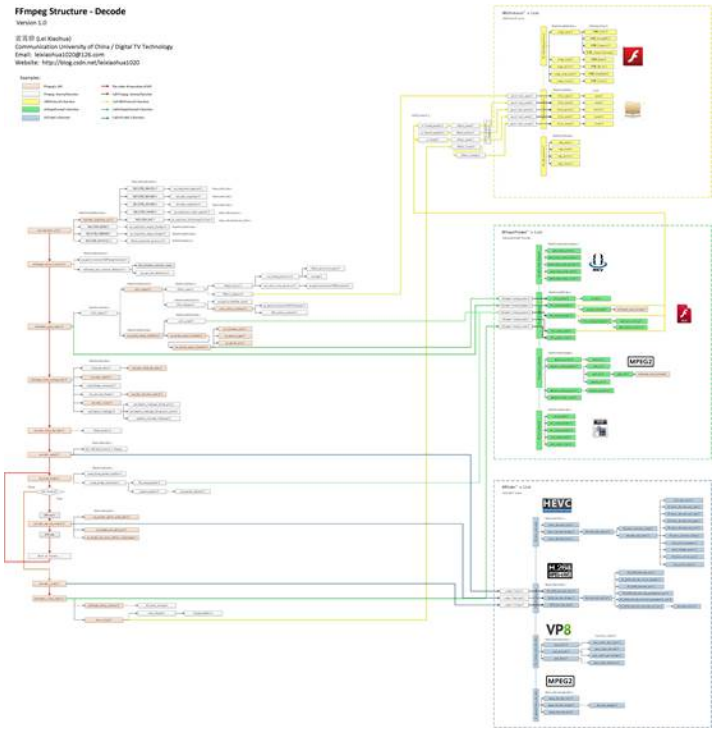
==========

近期研究了一下FFmpeg的内部的源代码。之前对于FFmpeg的研究主要在它的应用层面上，因此制作的很多示例程序都是调用的FFmpeg的API。但是一直感觉这样对FFmpeg的理解还是比较浅，所以打算剖析一下它的源代码，理一下它内部结构的“脉络”。但是有一个很难办的问题：FFmpeg自带的三个工程：ffplay，ffmpeg，ffprobe的代码量非常的大，其中包含了成百上千的API；而这些API背后又包含了大量的FFmpeg内部函数。如此一来，几乎是不可能理清他们之间的关系的。经过一番思考之后，打算选择FFmpeg编码和解码过程中的最核心的API进行分析。在编码或者解码的过程中，核心的API数量不多，一共大约10个左右，这样一来就可以剖析其内部的源代码了。

- FFmpeg解码过程核心的API可以参考：《 最简单的基于FFmpeg+SDL的视频播放器 》
- 编码过程核心的API可以参考：《 最简单的基于FFmpeg的视频编码器 》

FFmpeg源代码结构图-解码

首先呈现分析出来的FFmpeg源代码结构图。这张图的尺寸非常的大，尺寸大约有4000x4000，有点像一张地图（因此最好选择“查看更清晰的图片”之后，右键保存图片到本机之后再查看）。它表明了FFmpeg在解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，包含处理协议（Protocol）的功能。
- 绿色背景的函数：AVInputFormat结构体中的函数，包含处理封装格式（Format）的功能。
- 蓝色背景的函数：AVCodec结构体中的函数，包含了编解码器（Codec）的功能。

PS：URLProtocol，AVInputFormat，AVCodec在FFmpeg开始运行并且注册完组件之后，都会分别被连接成一个个的链表。因此实际上是有很多的URLProtocol，AVI

nputFormat，AVCodec的。图中画出了解码一个输入协议是“文件”（其实就是打开一个文件。“文件”也被当做是一种广义的协议），封装格式为FLV，视频编码格式是H.264的数据的函数调用关系。

区域

整个架构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，MPEGTS，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：

黑色箭头线：标志了函数之间的调用关系。

红色的箭头线：标志了解码的流程。

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AVInputFormat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

每个函数旁边标识了它所在的文件的路径。

此外，还有一点需要注意的是，一些API函数内部也调用了另一些API函数。也就是说，API函数并不一定全部都调用FFmpeg的内部函数，他也有可能调用其他的API函数。例如从图中可以看出来，avformat_close_input()调用了avformat_free_context()和avio_close()。这些在内部代码中被调用的API函数也标记为粉红色。

函数调用关系

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（FFmpeg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_context() 【函数简单分析】

1) av_malloc(sizeof(AVFormatContext))

2) avformat_get_context_defaults()

(a) av_opt_set_defaults()

3. avformat_open_input() 【函数简单分析】

1) init_input()

(a) avio_open2() 【函数简单分析】

a) ffurl_open()

i. ffurl_alloc()

| url_find_protocol()

| url_alloc_for_protocol()

ii. ffurl_connect()

URLProtocol->url_open()

b) ffio_fdopen()

i. av_malloc(buffer_size)

ii. avio_alloc_context()

| av_mallocz(sizeof(AVIOContext))

| ffio_init_context()

(b) av_probe_input_buffer2()

a) avio_read()

i. AVInputFormat->read_packet()

b) av_probe_input_format2()

c) av_probe_input_format3()

i. av_iformat_next()

ii. av_match_name()

iii. av_match_ext()

iv. AVInputFormat->read_probe()

2) AVInputFormat->read_header()

4. avformat_find_stream_info() 【函数简单分析】

1) find_decoder()

(a) avcodec_find_decoder()

2) avcodec_open2()

3) read_frame_internal()

4) try_decode_frame()

(a) avcodec_decode_video2()

5) avcodec_close()

6) estimate_timings()

(a) estimate_timings_from_pts()

(b) estimate_timings_from_bit_rate()

(c) update_stream_timings()

5. avcodec_find_decoder() 【函数简单分析】

1) find_encdec()

6. avcodec_open2() 【函数简单分析】

1) AVCodec->init()

7. av_read_frame() 【函数简单分析】

1) read_from_packet_buffer()

2) read_frame_internal()

(a) ff_read_packet()

a) AVInputFormat->read_packet()

(b) parse_packet()

a) av_parser_parse2()

8. avcodec_decode_video2() 【函数简单分析】

1) av_packet_split_side_data()

2) AVCodec-> decode()

3) av_frame_set_pkt_pos()

4) av_frame_set_best_effort_timestamp()

9. avcodec_close() 【函数简单分析】

1) AVCodec->close()

10. avformat_close_input() 【函数简单分析】

1) AVInputFormat->read_close()

2) avformat_free_context()

(a) ff_free_stream()

3) avio_close()

(a) avio_flush()

a) flush_buffer()

(b) ffurl_close()

a) ffurl_closep()

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（AVInputFormat封装格式处理函数）

AVInputFormat包含如下封装格式处理函数指针：

read_probe()：检查格式
read_header()：读取文件头
read_packet()：读取一帧数据
read_seek()：调整进度
read_close()：关闭

【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AVInputFormat结构体ff_flv_demuxer：

read_probe() -> flv_probe() –> probe()
read_header() -> flv_read_header() -> create_stream() -> avformat_new_stream()
read_packet() -> flv_read_packet()
read_seek() -> flv_read_seek()
read_close() -> flv_read_close()

MKV封装格式对应的AVInputFormat结构体ff_matroska_demuxer：

read_probe() -> matroska_probe()
read_header() -> matroska_read_header()
read_packet() -> matroska_read_packet()
read_seek() -> matroska_read_seek()
read_close() -> matroska_read_close()

MPEG2TS封装格式对应的AVInputFormat结构体ff_mpegts_demuxer：

read_probe() -> mpegts_probe()
read_header() -> mpegts_read_header()
read_packet() -> mpegts_read_packet()
read_close() -> mpegts_read_close()

AVI封装格式对应的AVInputFormat结构体ff_avi_demuxer：

read_probe() -> avi_probe()
read_header() -> avi_read_header()
read_packet() -> avi_read_packet()
read_seek() -> avi_read_seek()
read_close() -> avi_read_close()

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
decode()：解码一帧数据
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解码对应的AVCodec结构体ff_hevc_decoder：

init() -> hevc_decode_init()
decode() -> hevc_decode_frame() -> decode_nal_units()
close() -> hevc_decode_free()

H.264解码对应的AVCodec结构体ff_h264_decoder：

init() -> ff_h264_decode_init()
decode() -> h264_decode_frame() -> decode_nal_units()
close() -> h264_decode_end()

VP8解码（libVPX）对应的AVCodec结构体ff_libvpx_vp8_decoder：

init() -> vpx_init() -> vpx_codec_dec_init()
decode() -> vp8_decode() -> vpx_codec_decode(), vpx_codec_get_frame()
close() -> vp8_free() -> vpx_codec_destroy()

MPEG2解码对应的AVCodec结构体ff_mpeg2video_decoder：

init() -> mpeg_decode_init()
decode() -> mpeg_decode_frame()
close() -> mpeg_decode_end()

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

文章标签： FFmpeg 解码 架构 函数
个人分类： FFMPEG
所属专栏： FFmpeg