# FFmpeg解封装实验！

接下来开始给大家分享FFmpeg解封装演示，在写文章之前呢，还是说明一下，这篇文章我不会去把每个api接口都说明一下；我想准备一个专辑专门去讲，我会结合官网手册和FFmpeg源码接口说明去写，这目前是我的一个初步想法，具体行动的话，如果我开始写了第一篇关于api说明，那就说明计划已经执行起来了。。。。。

## **二、解封装流程：**

这里简单说一下，所谓的解封装就是把一个音视频文件拆开来，就像你把一个完整的玩具拆开来一样，然后可以对模块进行分析，当然与解封装对应就有封装了，这个不是本文的重点。

下面是解封装的流程：

1、注册相关模块（av\_register\_all; avformat\_network\_init）

2、根据即将输出的文件名、获取封装信息上下文AVFormatContext（avformat\_alloc\_output\_context2）

3、打开输出文件IO（avio\_open）

4、添加音视频流（avformat\_new\_stream）

5、封装文件头信息（avformat\_write\_header）

6、向文件中写入数据包，如果包含视频、音频等多个码流的数据包，则按照时间戳大小交织写入（av\_interleaved\_write\_frame）

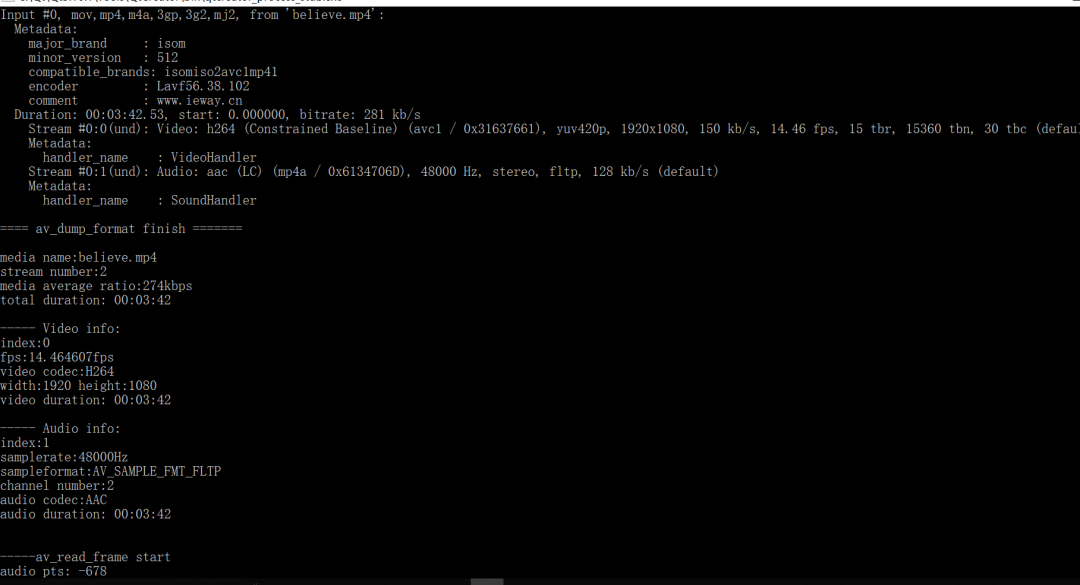
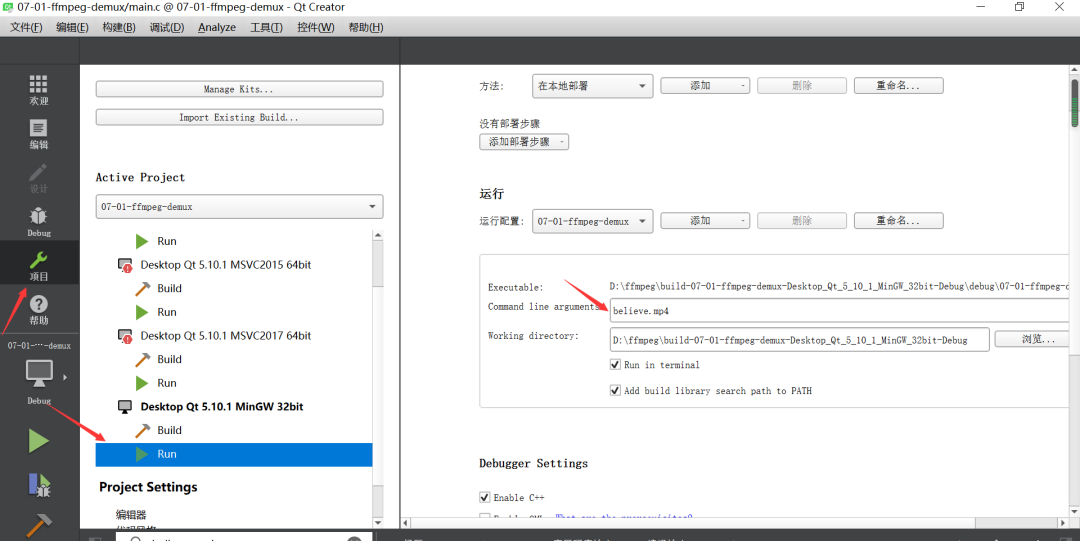
7、封装文件尾信息（av\_write\_trailer）

8、关闭操作

下面我直接把整个代码给出（这个代码测试mp4格式）：

*#include <stdio.h>*  
*#include <libavformat/avformat.h>*  
  
  
int main(int argc, char \*\*argv)  
{  
    //打开网络流。这里如果只需要读取本地媒体文件，不需要用到网络功能，可以不用加上这一句  
//    avformat\_network\_init();  
  
    const char \*default\_filename = "believe.mp4";  
  
    char \*in\_filename = NULL;  
  
    if(argv[1] == NULL)  
    {  
        in\_filename = default\_filename;  
    }  
    else  
    {  
        in\_filename = argv[1];  
    }  
    printf("in\_filename = %s\n", in\_filename);  
  
    //AVFormatContext是描述一个媒体文件或媒体流的构成和基本信息的结构体  
    AVFormatContext \*ifmt\_ctx = NULL;           // 输入文件的demux  
  
    int videoindex = -1;        // 视频索引  
    int audioindex = -1;        // 音频索引  
  
  
    // 打开文件，主要是探测协议类型，如果是网络文件则创建网络链接  
    int ret = avformat\_open\_input(&ifmt\_ctx, in\_filename, NULL, NULL);  
    if (ret < 0)  //如果打开媒体文件失败，打印失败原因  
    {  
        char buf[1024] = { 0 };  
        av\_strerror(ret, buf, sizeof(buf) - 1);  
        printf("open %s failed:%s\n", in\_filename, buf);  
        goto failed;  
    }  
  
    ret = avformat\_find\_stream\_info(ifmt\_ctx, NULL);  
    if (ret < 0)  //如果打开媒体文件失败，打印失败原因  
    {  
        char buf[1024] = { 0 };  
        av\_strerror(ret, buf, sizeof(buf) - 1);  
        printf("avformat\_find\_stream\_info %s failed:%s\n", in\_filename, buf);  
        goto failed;  
    }  
  
    //打开媒体文件成功  
    printf\_s("\n==== av\_dump\_format in\_filename:%s ===\n", in\_filename);  
    av\_dump\_format(ifmt\_ctx, 0, in\_filename, 0);  
    printf\_s("\n==== av\_dump\_format finish =======\n\n");  
    // url: 调用avformat\_open\_input读取到的媒体文件的路径/名字  
    printf("media name:%s\n", ifmt\_ctx->url);  
    // nb\_streams: nb\_streams媒体流数量  
    printf("stream number:%d\n", ifmt\_ctx->nb\_streams);  
    // bit\_rate: 媒体文件的码率,单位为bps  
    printf("media average ratio:%lldkbps\n",(int64\_t)(ifmt\_ctx->bit\_rate/1024));  
    // 时间  
    int total\_seconds, hour, minute, second;  
    // duration: 媒体文件时长，单位微妙  
    total\_seconds = (ifmt\_ctx->duration) / AV\_TIME\_BASE;  // 1000us = 1ms, 1000ms = 1秒  
    hour = total\_seconds / 3600;  
    minute = (total\_seconds % 3600) / 60;  
    second = (total\_seconds % 60);  
    //通过上述运算，可以得到媒体文件的总时长  
    printf("total duration: %02d:%02d:%02d\n", hour, minute, second);  
    printf("\n");  
    /\*  
     \* 老版本通过遍历的方式读取媒体文件视频和音频的信息  
     \* 新版本的FFmpeg新增加了函数av\_find\_best\_stream，也可以取得同样的效果  
     \*/  
    for (uint32\_t i = 0; i < ifmt\_ctx->nb\_streams; i++)  
    {  
        AVStream \*in\_stream = ifmt\_ctx->streams[i];// 音频流、视频流、字幕流  
        //如果是音频流，则打印音频的信息  
        if (AVMEDIA\_TYPE\_AUDIO == in\_stream->codecpar->codec\_type)  
        {  
            printf("----- Audio info:\n");  
            // index: 每个流成分在ffmpeg解复用分析后都有唯一的index作为标识  
            printf("index:%d\n", in\_stream->index);  
            // sample\_rate: 音频编解码器的采样率，单位为Hz  
            printf("samplerate:%dHz\n", in\_stream->codecpar->sample\_rate);  
            // codecpar->format: 音频采样格式  
            if (AV\_SAMPLE\_FMT\_FLTP == in\_stream->codecpar->format)  
            {  
                printf("sampleformat:AV\_SAMPLE\_FMT\_FLTP\n");  
            }  
            else if (AV\_SAMPLE\_FMT\_S16P == in\_stream->codecpar->format)  
            {  
                printf("sampleformat:AV\_SAMPLE\_FMT\_S16P\n");  
            }  
            // channels: 音频信道数目  
            printf("channel number:%d\n", in\_stream->codecpar->channels);  
            // codec\_id: 音频压缩编码格式  
            if (AV\_CODEC\_ID\_AAC == in\_stream->codecpar->codec\_id)  
            {  
                printf("audio codec:AAC\n");  
            }  
            else if (AV\_CODEC\_ID\_MP3 == in\_stream->codecpar->codec\_id)  
            {  
                printf("audio codec:MP3\n");  
            }  
            else  
            {  
                printf("audio codec\_id:%d\n", in\_stream->codecpar->codec\_id);  
            }  
            // 音频总时长，单位为秒。注意如果把单位放大为毫秒或者微妙，音频总时长跟视频总时长不一定相等的  
            if(in\_stream->duration != AV\_NOPTS\_VALUE)  
            {  
                int duration\_audio = (in\_stream->duration) \* av\_q2d(in\_stream->time\_base);  
                //将音频总时长转换为时分秒的格式打印到控制台上  
                printf("audio duration: %02d:%02d:%02d\n",  
                       duration\_audio / 3600, (duration\_audio % 3600) / 60, (duration\_audio % 60));  
            }  
            else  
            {  
                printf("audio duration unknown");  
            }  
  
            printf("\n");  
  
            audioindex = i; // 获取音频的索引  
        }  
        else if (AVMEDIA\_TYPE\_VIDEO == in\_stream->codecpar->codec\_type)  //如果是视频流，则打印视频的信息  
        {  
            printf("----- Video info:\n");  
            printf("index:%d\n", in\_stream->index);  
            // avg\_frame\_rate: 视频帧率,单位为fps，表示每秒出现多少帧  
            printf("fps:%lffps\n", av\_q2d(in\_stream->avg\_frame\_rate));  
            if (AV\_CODEC\_ID\_MPEG4 == in\_stream->codecpar->codec\_id) //视频压缩编码格式  
            {  
                printf("video codec:MPEG4\n");  
            }  
            else if (AV\_CODEC\_ID\_H264 == in\_stream->codecpar->codec\_id) //视频压缩编码格式  
            {  
                printf("video codec:H264\n");  
            }  
            else  
            {  
                printf("video codec\_id:%d\n", in\_stream->codecpar->codec\_id);  
            }  
            // 视频帧宽度和帧高度  
            printf("width:%d height:%d\n", in\_stream->codecpar->width,  
                   in\_stream->codecpar->height);  
            //视频总时长，单位为秒。注意如果把单位放大为毫秒或者微妙，音频总时长跟视频总时长不一定相等的  
            if(in\_stream->duration != AV\_NOPTS\_VALUE)  
            {  
                int duration\_video = (in\_stream->duration) \* av\_q2d(in\_stream->time\_base);  
                printf("video duration: %02d:%02d:%02d\n",  
                       duration\_video / 3600,  
                       (duration\_video % 3600) / 60,  
                       (duration\_video % 60)); //将视频总时长转换为时分秒的格式打印到控制台上  
            }  
            else  
            {  
                printf("video duration unknown");  
            }  
  
            printf("\n");  
            videoindex = i;  
        }  
    }  
  
    AVPacket \*pkt = av\_packet\_alloc();  
  
    int pkt\_count = 0;  
    int print\_max\_count = 10;  
    printf("\n-----av\_read\_frame start\n");  
    while (1)  
    {  
        ret = av\_read\_frame(ifmt\_ctx, pkt);  
        if (ret < 0)  
        {  
            printf("av\_read\_frame end\n");  
            break;  
        }  
  
        if(pkt\_count++ < print\_max\_count)  
        {  
            if (pkt->stream\_index == audioindex)  
            {  
                printf("audio pts: %lld\n", pkt->pts);  
                printf("audio dts: %lld\n", pkt->dts);  
                printf("audio size: %d\n", pkt->size);  
                printf("audio pos: %lld\n", pkt->pos);  
                printf("audio duration: %lf\n\n",  
                       pkt->duration \* av\_q2d(ifmt\_ctx->streams[audioindex]->time\_base));  
            }  
            else if (pkt->stream\_index == videoindex)  
            {  
                printf("video pts: %lld\n", pkt->pts);  
                printf("video dts: %lld\n", pkt->dts);  
                printf("video size: %d\n", pkt->size);  
                printf("video pos: %lld\n", pkt->pos);  
                printf("video duration: %lf\n\n",  
                       pkt->duration \* av\_q2d(ifmt\_ctx->streams[videoindex]->time\_base));  
            }  
            else  
            {  
                printf("unknown stream\_index:\n", pkt->stream\_index);  
            }  
        }  
  
        av\_packet\_unref(pkt);  
    }  
  
    if(pkt)  
        av\_packet\_free(&pkt);  
failed:  
    if(ifmt\_ctx)  
        avformat\_close\_input(&ifmt\_ctx);  
  
  
    getchar(); //加上这一句，防止程序打印完信息马上退出  
    return 0;  
}

首先演示一下mp4格式输出结果：



输出结果把音频和视频流分开来解析了，具体可以看上面的源码就知道了。

上面的代码注解已经非常详细了，理解起来应该不难，下面我debug一下可能有人看不懂的地方：

//视频总时长，单位为秒。注意如果把单位放大为毫秒或者微妙，音频总时长跟视频总时长不一定相等的  
            if(in\_stream->duration != AV\_NOPTS\_VALUE)  
            {  
                int duration\_video = (in\_stream->duration) \* av\_q2d(in\_stream->time\_base);  
                printf("video duration: %02d:%02d:%02d\n",  
                       duration\_video / 3600,  
                       (duration\_video % 3600) / 60,  
                       (duration\_video % 60)); //将视频总时长转换为时分秒的格式打印到控制台上  
            }

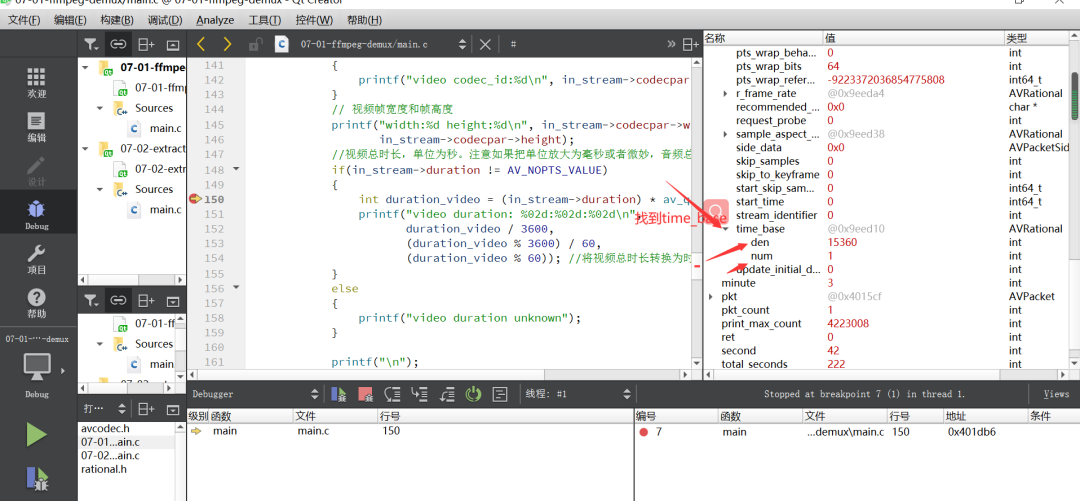
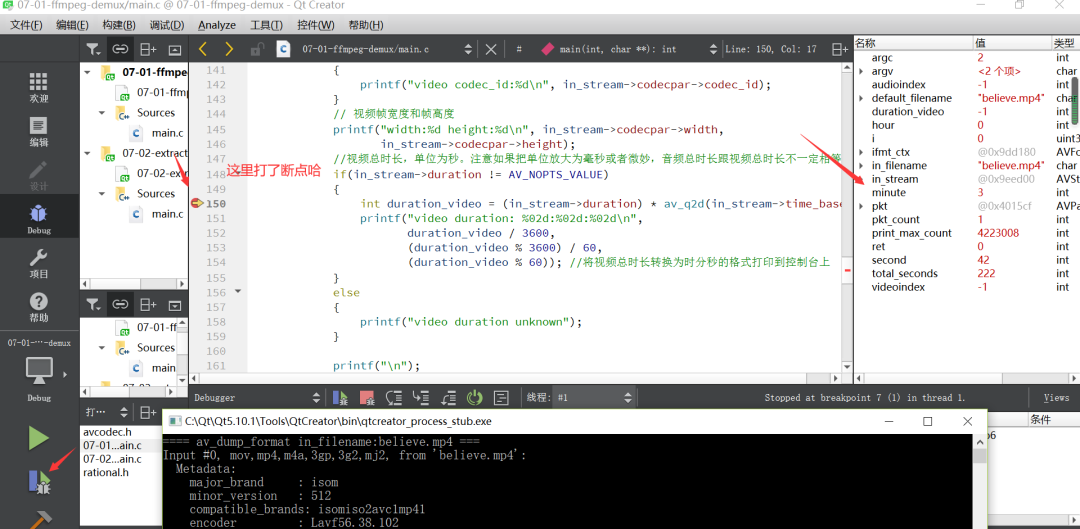
这段代码视频时间计算可能大家不好理解，特别是

av\_q2d(in\_stream->time\_base);

它是一个内联函数：

/\*\*  
 \* Convert an AVRational to a `double`.  
 \* @param a AVRational to convert  
 \* @return `a` in floating-point form  
 \* @see av\_d2q()  
 \*/  
static inline double av\_q2d(AVRational a){  
    return a.num / (double) a.den;  
}

那这里的num和den是多少呢？下面我们来看一下debug：



通过debug调试，我们可以发现den的值为15360，num的值为1，那么：

a.num / (double) a.den = 1 / 15360

这样就可以算出整个视频流的总时间了：

 int duration\_video = (in\_stream->duration) \* av\_q2d(in\_stream->time\_base);

