

## 原 FFMpeg源代码简单分析：av\_find\_decoder()和av\_find\_encoder()

2015年03月06日 12:40:06 阅读数：17041

=====

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](#)

本文记录FFmpeg的两个API函数：avcodec\_find\_encoder()和avcodec\_find\_decoder()。avcodec\_find\_encoder()用于查找FFmpeg的编码器，avcodec\_find\_decoder()用于查找FFmpeg的解码器。

avcodec\_find\_encoder()的声明位于libavcodec\avcodec.h，如下所示。

```
1.  /**
2.   * Find a registered encoder with a matching codec ID.
3.   *
4.   * @param id AVCodecID of the requested encoder
5.   * @return An encoder if one was found, NULL otherwise.
6.   */
7.  AVCodec *avcodec_find_encoder(enum AVCodecID id);
```

函数的参数是一个编码器的ID，返回查找到的编码器（没有找到就返回NULL）。

avcodec\_find\_decoder()的声明也位于libavcodec\avcodec.h，如下所示。

```
1.  /**
2.   * Find a registered decoder with a matching codec ID.
3.   *
4.   * @param id AVCodecID of the requested decoder
5.   * @return A decoder if one was found, NULL otherwise.
6.   */
7.  AVCodec *avcodec_find_decoder(enum AVCodecID id);
```

函数的参数是一个解码器的ID，返回查找到的解码器（没有找到就返回NULL）。

avcodec\_find\_encoder()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）](#)

avcodec\_find\_decoder()函数最典型的例子可以参考：

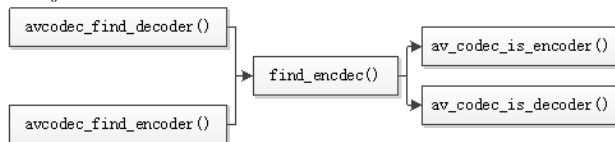
[最简单的基于FFMPEG+SDL的视频播放器 ver2（采用SDL2.0）](#)

其实这两个函数的实质就是遍历AVCodec链表并且获得符合条件的元素。有关AVCodec链表的建立可以参考文章：

[ffmpeg 源代码简单分析：av\\_register\\_all\(\)](#)

## 函数调用关系图

avcodec\_find\_encoder()和avcodec\_find\_decoder()的函数调用关系图如下所示。



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

## avcodec\_find\_encoder()



avcodec\_find\_encoder()的源代码位于libavcodec\utils.c，如下所示。

```
1.  AVCodec *avcodec_find_encoder(enum AVCodecID id)
2.  {
3.      return find_encdec(id, 1);
4.  }
```

从源代码可以看出avcodec\_find\_encoder()调用了一个find\_encdec()，注意它的第二个参数是0。  
下面我们看一下find\_encdec()的定义。

## find\_encdec()

find\_encdec()的源代码位于libavcodec\utils.c，如下所示。

```
[cpp]    
1. static AVCodec *first_avcodec;  
2.  
3. static AVCodec *find_encdec(enum AVCodecID id, int encoder)  
4. {  
5.     AVCodec *p, *experimental = NULL;  
6.     p = first_avcodec;  
7.     id= remap_deprecated_codec_id(id);  
8.     while (p) {  
9.         if ((encoder ? av_codec_is_encoder(p) : av_codec_is_decoder(p)) &&  
10.            p->id == id) {  
11.             if (p->capabilities & CODEC_CAP_EXPERIMENTAL && !experimental) {  
12.                 experimental = p;  
13.             } else  
14.                 return p;  
15.         }  
16.         p = p->next;  
17.     }  
18.     return experimental;  
19. }
```



find\_encdec()中有一个循环，该循环会遍历AVCodec结构的链表，逐一比较输入的ID和每一个编码器的ID，直到找到ID取值相等的编码器。

在这里有几点需要注意：



- (1) first\_avcodec是一个全局变量，存储AVCodec链表的第一个元素。
- (2) remap\_deprecated\_codec\_id()用于将一些过时的编码器ID映射到新的编码器ID。
- (3) 函数的第二个参数encoder用于确定查找编码器还是解码器。当该值为1的时候，用于查找编码器，此时会调用av\_codec\_is\_encoder()判断AVCodec是否为编码器；当该值为0的时候，用于查找解码器，此时会调用av\_codec\_is\_decoder()判断AVCodec是否为解码器。

## av\_codec\_is\_encoder()

av\_codec\_is\_encoder()是一个判断AVCodec是否为编码器的函数。如果是编码器，返回非0值，否则返回0。

```
[cpp]    
1. /**  
2.  * @return a non-zero number if codec is an encoder, zero otherwise  
3.  */  
4. int av_codec_is_encoder(const AVCodec *codec);
```



av\_codec\_is\_encoder()源代码很简单，如下所示。

```
[cpp]    
1. int av_codec_is_encoder(const AVCodec *codec)  
2. {  
3.     return codec && (codec->encode_sub || codec->encode2);  
4. }
```

从源代码可以看出，av\_codec\_is\_encoder()判断了一下AVCodec是否包含了encode2()或者encode\_sub()接口函数。

## av\_codec\_is\_decoder()

av\_codec\_is\_decoder()是一个判断AVCodec是否为解码器的函数。如果是解码器，返回非0值，否则返回0。

```
[cpp]    
1. /**  
2.  * @return a non-zero number if codec is a decoder, zero otherwise  
3.  */  
4. int av_codec_is_decoder(const AVCodec *codec);
```



av\_codec\_is\_decoder()源代码也很简单，如下所示。

```
[cpp]    
1. int av_codec_is_decoder(const AVCodec *codec)  
2. {  
3.     return codec && codec->decode;  
4. }
```

从源代码可以看出，av\_codec\_is\_decoder()判断了一下AVCodec是否包含了decode()接口函数。

## avcodec\_find\_decoder()

avcodec\_find\_decoder()的源代码位于libavcodec\utils.c, 如下所示。

```
[cpp]    
1. AVCodec *avcodec_find_decoder(enum AVCodecID id)  
2. {  
3.     return find_encdec(id, 0);  
4. }
```

可以看出avcodec\_find\_decoder()同样调用了find\_encdec(), 只是第2个参数设置为0。因此不再详细分析。

**雷霄骅**

**leixiaohua1020@126.com**

**<http://blog.csdn.net/leixiaohua1020>**

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/44084557>

文章标签： [FFmpeg](#) [源代码](#) [编码器](#) [解码器](#)

个人分类： [FFMPEG](#)

所属专栏： [FFmpeg](#)

此PDF由spygg生成, 请尊重原作者版权!!!

我的邮箱:liushidc@163.com