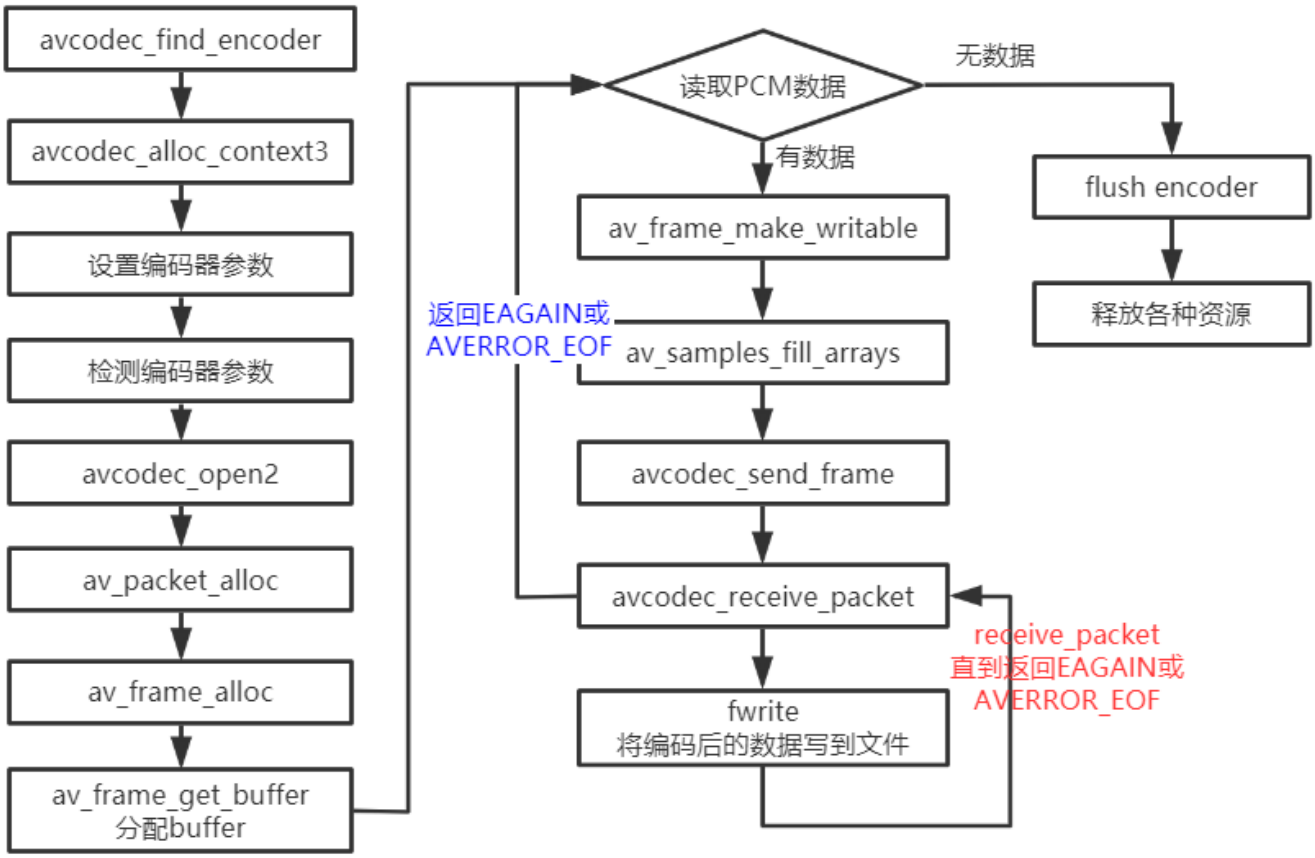
音频编码实战

FFmpeg流程

从本地文件读取PCM数据进行AAC格式编码，然后将编码后的AAC数据存储到本地文件。
示例的流程如下所示。



关键函数说明：

- avcodec_find_encoder：根据指定的AVCodecID查找注册的编码器。
- avcodec_alloc_context3：为AVCodecContext分配内存。
- avcodec_open2：打开编码器。
- avcodec_send_frame：将AVFrame非压缩数据给编码器。
- avcodec_receive_packet：获取到编码后的AVPacket数据，收到的packet需要自己释放内存。
- av_frame_get_buffer: 为音频或视频帧分配新的buffer。在调用这个函数之前，必须在AVFame上设置好以下属性：format(视频为像素格式，音频为样本格式)、nb_samples(样本个数，针对音频)、channel_layout(通道类型，针对音频)、width/height(宽高，针对视频) 。
- av_frame_make_writable：确保AVFrame是可写的，使用av_frame_make_writable()的问题是，在最坏的情况下，它会在您使用encode再次更改整个输入frame之前复制它. 如果frame不可写，av_frame_make_writable()将分配新的缓冲区，并复制这个输入input frame数据，避免和编码器需要缓存该帧时造成冲突。
- av_samples_fill_arrays 填充音频帧

对于 flush encoder的操作：

编码器通常的冲洗方法：调用一次 avcodec_send_frame(NULL)(返回成功)，然后不停调用 avcodec_receive_packet() 直到其返回 AVERROR_EOF，取出所有缓存帧，

avcodec_receive_packet() 返回 AVERROR_EOF 这一次是没有有效数据的，仅仅获取到一个结束标志

PCM样本格式

PCM(Pulse Code Modulation，脉冲编码调制)音频数据是未经压缩的音频采样数据裸流，它是由模拟信号经过采样、量化、编码转换成的标准数字音频数据。
描述PCM数据的6个参数：

1. Sample Rate：采样频率。8kHz(电话)、44.1kHz(CD)、48kHz(DVD)。
2. Sample Size：量化位数。通常该值为16-bit。
3. Number of Channels：通道个数。常见的音频有立体声(stereo)和单声道(mono)两种类型，立体声包含左声道和右声道。另外还有环绕立体声等其它不太常用的类型。
4. Sign：表示样本数据是否是有符号位，比如用一字节表示的样本数据，有符号的话表示范围为-128 ~ 127，无符号是0 ~ 255。有符号位16bits数据取值范围为-32768~32767。
5. Byte Ordering：字节序。字节序是little-endian还是big-endian。通常均为little-endian。字节序说明见第4节。
6. Integer Or Floating Point：整形或浮点型。大多数格式的PCM样本数据使用整形表示，而在一些对精度要求高的应用方面，使用浮点类型表示PCM样本数据（浮点数 float值域为 [-1.0, 1.0]）。

推荐的PCM数据播放工具：

- ffplay，使用示例如下：

```
//播放格式为f32le，双声道，采样频率48000Hz的PCM数据
ffplay -f f32le -ac 2 -ar 48000 pcm_audio
```

- Audacity：一款免费开源的跨平台音频处理软件。
- Adobe Auditon。导入原始数据，打开的时候需要选择采样率、格式和字节序。

FFmpeg支持的PCM数据格式

使用ffmpeg -formats命令，获取ffmpeg支持的音视频格式，其中我们可以找到支持的PCM格式。

```
DE alaw            PCM A-law
 DE f32be           PCM 32-bit floating-point big-endian
 DE f32le           PCM 32-bit floating-point little-endian
 DE f64be           PCM 64-bit floating-point big-endian
 DE f64le           PCM 64-bit floating-point little-endian
 DE mulaw           PCM mu-law
```

```
 7  DE s16be           PCM signed 16-bit big-endian
 8  DE s16le           PCM signed 16-bit little-endian
 9  DE s24be           PCM signed 24-bit big-endian
10  DE s24le           PCM signed 24-bit little-endian
11  DE s32be           PCM signed 32-bit big-endian
12  DE s32le           PCM signed 32-bit little-endian
13  DE s8              PCM signed 8-bit
14  DE u16be           PCM unsigned 16-bit big-endian
15  DE u16le           PCM unsigned 16-bit little-endian
16  DE u24be           PCM unsigned 24-bit big-endian
17  DE u24le           PCM unsigned 24-bit little-endian
18  DE u32be           PCM unsigned 32-bit big-endian
19  DE u32le           PCM unsigned 32-bit little-endian
20  DE u8              PCM unsigned 8-bit
```

s是有符号，u是无符号，f是浮点数。
be是大端，le是小端。

FFmpeg中Packed和Planar的PCM数据区别

FFmpeg中音视频数据基本上都有Packed和Planar两种存储方式，对于双声道音频来说，Packed方式为两个声道的数据交错存储；Planar方式为两个声道分开存储。假设一个L/R为一个采样点，数据存储的方式如下所示：

- Packed: L R L R L R L R
- Planar: L L L L ... R R R R...

packed格式

```
1 AV_SAMPLE_FMT_U8,          ///< unsigned 8 bits
2 AV_SAMPLE_FMT_S16,         ///< signed 16 bits
3 AV_SAMPLE_FMT_S32,         ///< signed 32 bits
4 AV_SAMPLE_FMT_FLT,         ///< float
5 AV_SAMPLE_FMT_DBL,         ///< double
```

只能保存在AVFrame的uint8_t *data[0];

音频保持格式如下：

> LRLRLR ...

planar格式

planar为FFmpeg内部存储音频使用的采样格式，所有的Planar格式后面都有字母P标识。

```
AV_SAMPLE_FMT_U8P,          ///< unsigned 8 bits, planar
AV_SAMPLE_FMT_S16P,         ///< signed 16 bits, planar
AV_SAMPLE_FMT_S32P,         ///< signed 32 bits, planar
AV_SAMPLE_FMT_FLTP,         ///< float, planar
AV_SAMPLE_FMT_DBLP,         ///< double, planar
AV_SAMPLE_FMT_S64,          ///< signed 64 bits
AV_SAMPLE_FMT_S64P,         ///< signed 64 bits, planar
```

plane 0: LLLLLLLLLLLLLLLLLLLLLLLLL...
plane 1: RRRRRRRRRRRRRRRRRRR....

plane 0对于uint8_t *data[0];
plane 1对于uint8_t *data[1];

FFmpeg默认的AAC编码器不支持AV_SAMPLE_FMT_S16格式的编码，只支持AV_SAMPLE_FMT_FLTP，这种格式是按平面存储，样点是float类型，所谓平面也就是

> 每个声道单独存储，比如左声道存储到data[0]中，右声道存储到data[1]中。

FFmpeg音频**解码后**和**编码前**的数据是存放在AVFrame结构中的。

- Packed格式，frame.data[0]或frame.extended_data[0]包含所有的音频数据中。
- Planar格式，frame.data[i]或者frame.extended_data[i]表示第i个声道的数据（假设声道0是第一个），AVFrame.data数组大小固定为8，如果声道数超过8，需要从frame.extended_data获取声道数据。

补充说明

- Planar模式是ffmpeg内部存储模式，我们实际使用的音频文件都是Packed模式的。
- FFmpeg解码不同格式的音频输出的音频采样格式不是一样。测试发现，其中AAC解码输出的数据为浮点型的 AV_SAMPLE_FMT_FLTP 格式，MP3解码输出的数据为 AV_SAMPLE_FMT_S16P 格式（使

用的mp3文件为16位深）。具体采样格式可以查看解码后的AVFrame中的format成员或编解码器的AVCodecContext中的sample_fmt成员。

- Planar或者Packed模式直接影响到保存文件时写文件的操作，操作数据的时候一定要先检测音频采样格式。

PCM字节序

谈到字节序的问题，必然牵涉到两大CPU派系。那就是Motorola的PowerPC系列CPU和Intel的x86系列CPU。PowerPC系列采用big endian方式存储数据，而x86系列则采用little endian方式存储数据。那么究竟什么是big endian，什么又是little endian?
big endian是指低地址存放最高有效字节（MSB，Most Significant Bit），而little endian则是低地址存放最低有效字节（LSB，Least Significant Bit）。
下面用图像加以说明。比如数字0x12345678在两种不同字节序CPU中的存储顺序如下所示：
Big Endian
低地址 高地址

-->
| 12 | 34 | 56 | 78 |
Little Endian
低地址 高地址

-->
| 78 | 56 | 34 | 12 |
所有网络协议都是采用big endian的方式来传输数据的。所以也把big endian方式称之为网络字节序。当两台采用不同字节序的主机通信时，在发送数据之前都必须经过字节序的转换成为网络字节序后再进行传输。

作业

avcodec_receive_packet 不同的返回值代表什么含义；读取的packet如果要放到队列里面那应该怎么放到队列?