**FFmpeg数据流及codec添加**

FFmpeg的数据流程（以ffplay为例）

1. Ffplay 注册codecs，protocols等并解析options;
2. 启动decode\_thread线程;
3. 调用av\_open\_input\_file：探测文件格式；
4. 调用av\_open\_input\_stream：打开IO stream；
5. 调用AVInputFormat的read\_header()获得具体文件格式的metadata信息；
6. 调用av\_find\_stream\_info: 设置codec参数；
7. 调用stream\_component\_open：查找并打开decoder并设置回调sdl\_audio\_callback；
8. 调用audio\_decode\_frame：解码一帧数据；
9. 启动refresh\_thread线程：负责触发FF\_REFRESH\_EVENT事件并由event\_loop()作相应的处理；
10. 调用av\_read\_frame：读取数据包；
11. 将读取的数据包压入相应的包队列；
12. 进入event\_loop()事件循环；

FFmpeg的Codec组织方法

Ffmpeg使用AVCodec结构体描述一个codec，下面是AVCodec结构体：

typedef struct **AVCodec** {

const char \*name; //全局唯一的encoders或者decoders命名；

enum AVMediaType type;

enum CodecID id; //codec 的ID

int priv\_data\_size;

int (\*init)(AVCodecContext \*); //codec的初始化；

int (\*encode)(AVCodecContext \*, uint8\_t \*buf, int buf\_size, void \*data); // codec的编码函数入口；

int (\*close)(AVCodecContext \*); // codec的关闭函数入口；

int (\*decode)(AVCodecContext \*, void \*outdata, int \*outdata\_size, AVPacket \*avpkt); //codec的解码函数入口

int capabilities;

struct AVCodec \*next;

void (\*flush)(AVCodecContext \*); //codec 的清空buffer的函数入口

const AVRational \*supported\_framerates;

const enum PixelFormat \*pix\_fmts;

const char \*long\_name;

const int \*supported\_samplerates;

const enum SampleFormat \*sample\_fmts;

const int64\_t \*channel\_layouts;

} **AVCodec**;

Ffmpeg使用AVCodecContext结构体作为对codec调用接口的参数类型，传递codec需要的参数等信息，对于AVCodecContext结构体的具体内容可参见libavcodec/avcodec.h文件。

FFmpeg添加codec的步骤：(以wma为例)

1. 在libavcodec/avcodec.h 的CodecID 枚举中添加wma的codec id 比如 CODEC\_ID\_WMAV1
2. 在libavcodec/allcodecs.c的avcodec\_register\_all()函数中添加wma 解码器的注册比如： REGISTER\_ENCDEC(WMAV1, wmav1);
3. 在libavformat/xxx.c文件中添加对于可包含wma数据的文件格式对于wmav1的识别；（此处的实现并不唯一，具体可参见ff\_codec\_wav\_tags）
4. 创建wma.c 文件并实现 init, close, decode, flush函数，并定义一个AVCodec 变量，比如：

AVCodec wmav1\_decoder =

{

"wmav1", //name

AVMEDIA\_TYPE\_AUDIO,

CODEC\_ID\_WMAV1, //codec id

sizeof(WMACodecContext), //wma 结构体

wma\_decode\_init, // init 函数入口

NULL,

ff\_wma\_end, //close 函数入口

wma\_decode\_superframe, // decode 函数入口

.flush=flush, //flush 函数入口

};

针对目前将wma,aac, mp3等定点解码库替换FFmpeg现有的实现，只需要重新实现init,close,decode,flush函数，实现的具体方法与各个定点解码库现有的实现有关，只要保证对传递的AVCodecContext类型的参数做正确的处理即可。