# 环境搭建

需要完整安装ffmpeg

## 下载源码包

## 安装

# 常用的命令

https://ffmpeg.org/documentation.html

## 分离视频音频流

-vn和-an分别是屏蔽视频流和屏蔽音频流，分别对源文件处理一次即可得到分离的音频和视频。

ffmpeg -i input\_file -vcodec copy -an output\_file\_video　　//分离视频流

ffmpeg -i input\_file -acodec copy -vn output\_file\_audio　　//分离音频流

**范例 2-1 分离音频，分离视频**

ffmpeg -i source.200kbps.768x320.flv -acodec copy -vn audio.aac

ffmpeg -i source.200kbps.768x320.flv -vcodec copy -an video.mp4

解到容器里

测试分离的文件

ffplay -stats -f h264

ffplay -stats -f h264

## 视频解复用

ffmpeg -i test.mp4 -vcodec copy -an -f m4v test.264

ffmpeg -i test.avi -vcodec copy -an -f m4v test.264

范例2-2 视频解复用

ffmpeg -i source.200kbps.768x320.flv -f h264 video.h264

## 视频转码

ffmpeg -i test.mp4 -vcodec h264 -s 352\*278 -an -f m4v test.264 //转码为码流原始文件

ffmpeg -i test.mp4 -vcodec h264 -bf 0 -g 25 -s 352\*278 -an -f m4v test.264 //转码为码流原始文件

ffmpeg -i test.avi -vcodec mpeg4 -vtag xvid -qsame test\_xvid.avi //转码为封装文件

//-bf B帧数目控制，-g 关键帧间隔控制，-s 分辨率控制

## 视频封装

ffmpeg -i video\_file -i audio\_file -vcodec copy -acodec copy output\_file

范例

ffmpeg -i source.200kbps.768x320.flv -vcodec copy -acodec copy video.mp4

ffmpeg -i source.200kbps.768x320.flv -vcodec copy -acodec copy video.mkv

当转为ts和avi格式时报错

ffmpeg -i source.200kbps.768x320.flv -vcodec copy -acodec copy video.ts

报错

[mpegts @ 0xc724e0] H.264 bitstream malformed, no startcode found, use the video bitstream filter 'h264\_mp4toannexb' to fix it ('-bsf:v h264\_mp4toannexb' option with ffmpeg)

av\_interleaved\_write\_frame(): Invalid data found when processing input

改为

ffmpeg -i source.200kbps.768x320.flv -vcodec copy -acodec copy -bsf video.ts

报错

At least one output file must be specified

## 视频剪切

ffmpeg -i test.avi -r 1 -f image2 image-%3d.jpeg //提取图片

ffmpeg -ss 0:1:30 -t 0:0:20 -i input.avi -vcodec copy -acodec copy output.avi //剪切视频

//-r 提取图像的频率，-ss 开始时间，-t 持续时间

6.视频录制

ffmpeg -i rtsp://192.168.3.205:5555/test -vcodec copy out.avi

7.YUV序列播放

ffplay -f rawvideo -video\_size 1920x1080 input.yuv

8.YUV序列转AVI

ffmpeg -s w\*h -pix\_fmt yuv420p -i input.yuv -vcodec mpeg4 output.avi

1、ffmpeg使用语法

命令格式：

ffmpeg -i [输入文件名] [参数选项] -f [格式] [输出文件]

ffmpeg [[options][`-i' input\_file]]... {[options] output\_file}...

1、参数选项：

(1) -an: 去掉音频

(2) -acodec: 音频选项， 一般后面加copy表示拷贝

(3) -vcodec:视频选项，一般后面加copy表示拷贝

2、格式：

(1) h264: 表示输出的是h264的视频裸流

(2) mp4: 表示输出的是mp4的视频

(3)mpegts: 表示ts视频流

如果没有输入文件，那么视音频捕捉（只在Linux下有效，因为Linux下把音视频设备当作文件句柄来处理）就会起作用。作为通用的规则，选项一般用于下一个特定的文件。如果你给 -b 64选项，改选会设置下一个视频速率。对于原始输入文件，格式选项可能是需要的。缺省情况下，ffmpeg试图尽可能的无损转换，采用与输入同样的音频视频参数来输出。（by ternence.hsu）

2、视频转换

H264视频转ts视频流

ffmpeg -i test.h264 -vcodec copy -f mpegts test.ts

H264视频转mp4

ffmpeg -i test.h264 -vcodec copy -f mp4 test.mp4

ts视频转mp4

ffmpeg -i test.ts -acodec copy -vcodec copy -f mp4 test.mp4

mp4视频转flv

ffmpeg -i test.mp4 -acodec copy -vcodec copy -f flv test.flv

转换文件为3GP格式

ffmpeg -y -i test.mpeg -bitexact -vcodec h263 -b 128 -r 15 -s 176x144 -acodec aac -ac 2 -ar 22500 -ab 24 -f 3gp test.3gp

转换文件为3GP格式 v2

ffmpeg -y -i test.wmv -ac 1 -acodec libamr\_nb -ar 8000 -ab 12200 -s 176x144 -b 128 -r 15 test.3gp

使用 ffmpeg 编码得到高质量的视频

ffmpeg.exe -i "D:\Video\Fearless\Fearless.avi" -target film-dvd -s 720x352 -padtop 64 -padbottom 64 -maxrate 7350000 -b 3700000 -sc\_threshold 1000000000 -trellis -cgop -g 12 -bf 2 -qblur 0.3 -qcomp 0.7 -me full -dc 10 -mbd 2 -aspect 16:9 -pass 2 -passlogfile "D:\Video\ffmpegencode" -an -f mpeg2video "D:\Fearless.m2v"

转换指定格式文件到FLV格式

ffmpeg.exe -i test.mp3 -ab 56 -ar 22050 -b 500 -r 15 -s 320x240 f:\test.flv

ffmpeg.exe -i test.wmv -ab 56 -ar 22050 -b 500 -r 15 -s 320x240 f:\test.flv

转码解密的VOB

ffmpeg -i snatch\_1.vob -f avi -vcodec mpeg4 -b 800 -g 300 -bf 2 -acodec mp3 -ab 128 snatch.avi

（上面的命令行将vob的文件转化成avi文件，mpeg4的视频和mp3的音频。注意命令中使用了B帧，所以mpeg4流是divx5兼容的。GOP大小是300意味着29.97帧频下每10秒就有INTRA帧。该映射在音频语言的DVD转码时候尤其有用，同时编码到几种格式并且在输入流和输出流之间建立映射）

转换文件为3GP格式

ffmpeg -i test.avi -y -b 20 -s sqcif -r 10 -acodec amr\_wb -ab 23.85 -ac 1 -ar 16000 test.3gp

（如果要转换为3GP格式，则ffmpeg在编译时必须加上-enable-amr\_nb -enable-amr\_wb，详细内容可参考：转换视频为3GPP格式）

转换文件为MP4格式（支持iPhone/iTouch）

ffmpeg -y -i input.wmv -f mp4 -async 1-s 480x320 -acodec libfaac -vcodec libxvid -qscale 7 -dts\_delta\_threshold 1 output.mp4

ffmpeg -y -i source\_video.avi input -acodec libfaac -ab 128000 -vcodec mpeg4 -b 1200000 -mbd 2 -flags +4mv+trell -aic 2 -cmp 2 -subcmp 2 -s 320x180 -title X final\_video.mp4

将一段音频与一段视频混合

ffmpeg -i son.wav -i video\_origine.avi video\_finale.mpg

将一段视频转换为DVD格式

ffmpeg -i source\_video.avi -target pal-dvd -ps 2000000000 -aspect 16:9 finale\_video.mpeg

（target pal-dvd : Output format ps 2000000000 maximum size for the output file, in bits (here, 2 Gb) aspect 16:9 : Widescreen）

转换一段视频为DivX格式

ffmpeg -i video\_origine.avi -s 320x240 -vcodec msmpeg4v2 video\_finale.avi

Turn X images to a video sequence

ffmpeg -f image2 -i image%d.jpg video.mpg

（This command will transform all the images from the current directory (named image1.jpg, image2.jpg, etc...) to a video file named video.mpg.）

Turn a video to X images

ffmpeg -i video.mpg image%d.jpg

（This command will generate the files named image1.jpg, image2.jpg, ... ；The following image formats are also availables : PGM, PPM, PAM, PGMYUV, JPEG, GIF, PNG, TIFF, SGI.）

使用ffmpeg录像屏幕(仅限Linux平台)

ffmpeg -vcodec mpeg4 -b 1000 -r 10 -g 300 -vd x11:0,0 -s 1024x768 ~/test.avi

（-vd x11:0,0 指录制所使用的偏移为 x=0 和 y=0，-s 1024×768 指录制视频的大小为 1024×768。录制的视频文件为 test.avi，将保存到用户主目录中；如果你只想录制一个应用程序窗口或者桌面上的一个固定区域，那么可以指定偏移位置和区域大小。使用xwininfo -frame命令可以完成查找上述参数。）

重新调整视频尺寸大小(仅限Linux平台)

ffmpeg -vcodec mpeg4 -b 1000 -r 10 -g 300 -i ~/test.avi -s 800×600 ~/test-800-600.avi

把摄像头的实时视频录制下来，存储为文件(仅限Linux平台)

ffmpeg -f video4linux -s 320\*240 -r 10 -i /dev/video0 test.asf

使用ffmpeg压制H.264视频

ffmpeg -threads 4 -i INPUT -r 29.97 -vcodec libx264 -s 480x272 -flags +loop -cmp chroma -deblockalpha 0 -deblockbeta 0 -crf 24 -bt 256k -refs 1 -coder 0 -me umh -me\_range 16 -subq 5 -partitions parti4x4+parti8x8+partp8x8 -g 250 -keyint\_min 25 -level 30 -qmin 10 -qmax 51 -trellis 2 -sc\_threshold 40 -i\_qfactor 0.71 -acodec libfaac -ab 128k -ar 48000 -ac 2 OUTPUT

（使用该指令可以压缩出比较清晰，而且文件转小的H.264视频文件）

3、网络推送

udp视频流的推送

ffmpeg -re -i 1.ts -c copy -f mpegts udp://192.168.0.106:1234

4、视频拼接

裸码流的拼接，先拼接裸码流，再做容器的封装

ffmpeg -i "concat:test1.h264|test2.h264" -vcodec copy -f h264 out12.h264

5、图像相关

截取一张352x240尺寸大小的，格式为jpg的图片

ffmpeg -i test.asf -y -f image2 -t 0.001 -s 352x240 a.jpg

把视频的前30帧转换成一个Animated Gif

ffmpeg -i test.asf -vframes 30 -y -f gif a.gif

截取指定时间的缩微图,-ss后跟的时间单位为秒

ffmpeg -i test.avi -y -f image2 -ss 8 -t 0.001 -s 350x240 test.jpg

6、音频处理

转换wav到mp2格式

ffmpeg -i /tmp/a.wav -ab 64 /tmp/a.mp2 -ab 128 /tmp/b.mp2 -map 0:0 -map 0:0

（上面的命令行转换一个64Kbits 的a.wav到128kbits的a.mp2 ‘-map file:index’在输出流的顺序上定义了哪一路输入流是用于每一个输出流的。）

7、切割ts分片

ffmpeg -i input.mp4 -c:v libx264 -c:a aac -strict -2 -f hls -hls\_list\_size 6 -hls\_time 5 output1.m3u8

## 播放

## 转码

## 添加水印

# 编程范例

演示范例的开发原则，以总分总的方式进行，先演示一个简单播放器的开发；然后演示音视频的编解码，音频的重采样（音效处理），视频画面处理（尺寸改变、画面改变）；最后使用开发一个完整的播放器。

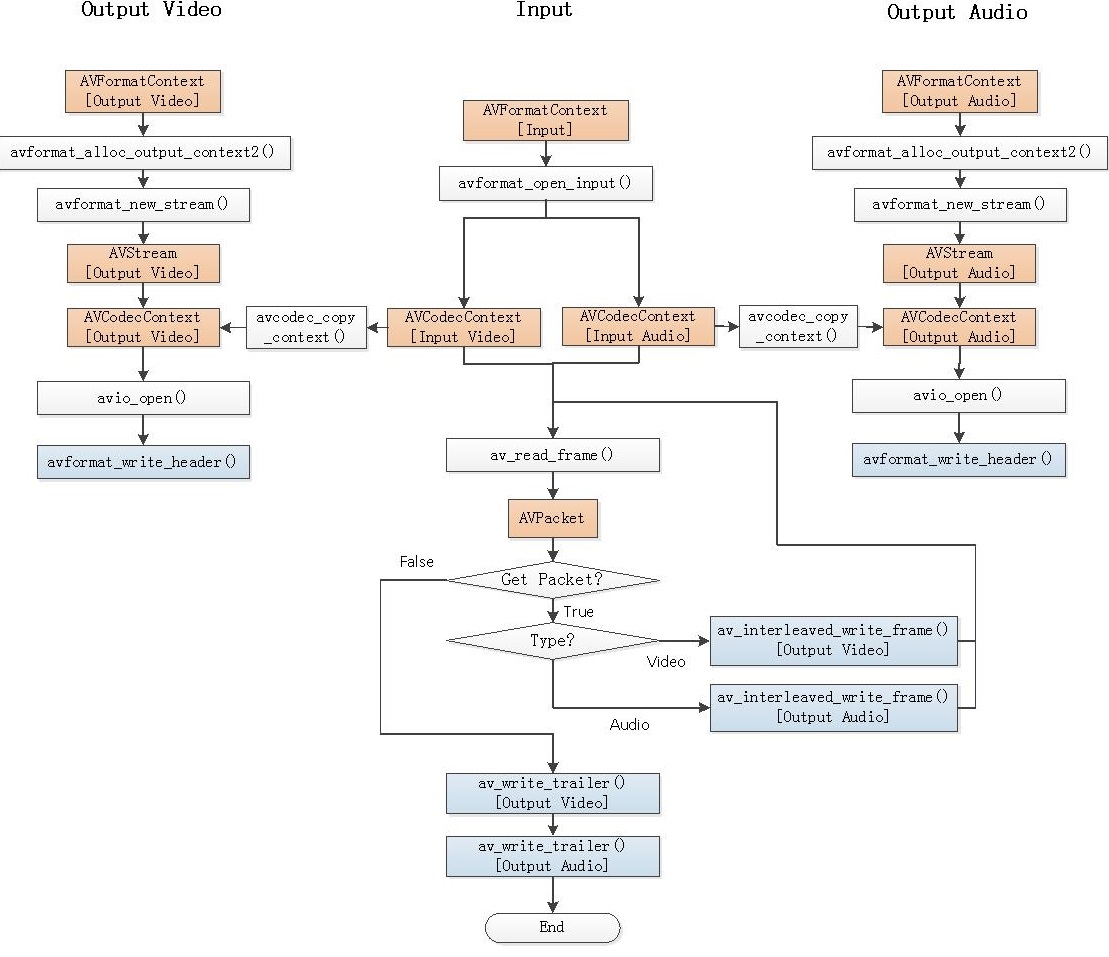
1. 简单播放器
2. 解封装音视频分离
3. 解码H264，并存储为yuv格式和rgb格式
4. 解码AAC，并存储为PCM
5. Yuv编码为H264
6. PCM编码为AAC
7. 将H264和AAC封装为FLV格式
8. 将YUV和PCM编码后封装为FLV格式
9. 将FLV格式文件转封装为MP4/TS/MKV
10. 视频改变缩放比例,并保存为原始格式
11. 音频图像处理（调整亮度、对比度等）
12. 音频重采样
13. 音频音效处理（重低音）
14. 视频转码H264转MP4
15. 视频转码MP4转码H264
16. 音频MP3转码AAC
17. 音频AAC转码MP3
18. 增加水印
19. 为视频添加/更换背景音乐
20. 两段视频做拼接，不同封装不同的解码器，输出指定为FLV封装，H264+AAC编码
21. 实现画中画效果并重新编码。

## 简单播放器

视频显示独立，音频播放独立。方便后续的范例使用。

## 解封装音视频分离

### 框架



雷大神

### 代码分析

### 验证分离后的文件

使用ffplay进行播放

播放H264文件

ffplay source.200kbps.768x320.h264

播放AAC文件

ffprobe source.200kbps.768x320.aac

### 参考

1. FFmpeg解封装音视频分离 <https://www.cnblogs.com/ht-beyond/p/5225664.html.>

## 解码H264，并存储为yuv格式和rgb格式

将3-2分离出来的文件source.200kbps.768x320.h264作为操作对象。

本文存储yuv和rgb的时候，都对图像做了缩放的操作。

### 框架

### 代码分析

Yuv和rgb的数据存储格式存在不同。

### 结果验证

使用ffplay进行播放

ffplay -f rawvideo -video\_size 768x320 768x320.yuv

ffplay -f rawvideo -pixel\_format rgb24 -video\_size 1920x1080 e:\workspace\encoder\output\test\_cff.rgb

播放RBG文件

ffplay -f rawvideo -pixel\_format rgb24 -video\_size 240x135 240x135.rgb

播放YUV文件

ffplay -f rawvideo -pixel\_format yuv420p -video\_size 320x180 320x180.yuv

### 参考

1. 将h264编码的视频流保存为图片http://blog.csdn.net/oldmtn/article/details/46742555

## 解码AAC，并存储为PCM

范例 3-4-decode\_aac

解码aac，使用3-2分离出来的aac文件

解码后的文件保存为

### 结果验证

{ AV\_SAMPLE\_FMT\_U8, "u8", "u8" },

{ AV\_SAMPLE\_FMT\_S16, "s16be", "s16le" },

{ AV\_SAMPLE\_FMT\_S32, "s32be", "s32le" },

{ AV\_SAMPLE\_FMT\_FLT, "f32be", "f32le" },

{ AV\_SAMPLE\_FMT\_DBL, "f64be", "f64le" },

ffplay -ar 44100 -channels 2 -f s16le -i xxx.pcm

**播放f32lex2.pcm**

ffplay -ar 44100 -channels 2 -f f32le -i f32lex2.pcm

**播放s16lex2.pcm**

ffplay -ar 44100 -channels 2 -f s16le s16lex2.pcm

## Yuv编码为H264（暂时不处理）

## PCM编码为AAC（编码）

范例 3-6-encode\_aac

使用范例3-4-decode\_aac产生的PCM数据作为实验，分不同的PCM编码进行编码，同时编码两个文件。

### 框架

pcm是最原始的音频编码格式，这种编码是无损的。同时意味着存储这种数据的文件将会很庞大，因此必须进行压缩。pcm是音频的编码格式，它不是文件的封装格式，上一节我们录制的声音存储在一个.pcm为后缀的文件中，这只是我们愿意这么做而已，你完全可以不这么做，这没有关系。

aac既是一种文件的封装格式，又是音频的编码格式。一aac为封装格式的文件，以.aac为后缀。aac封装格式一般内部的音频数据编码格式也为aac。

音频编码和视频编码的流程基本一致，而视频编码我们在前面已经做过了。因此，关于程序的流程就没有太多需要废话的了。下面介绍几个音频相关的参数，这几个参数是编码器进行编码所必需的。

我们总共需要设置四个参数即可：

1.sample\_rate

codecContext->sample\_rate = frame->sample\_rate;

sample\_rate指的是采样率。也就是我们一秒钟采集多少次声音样本。

2.frame->channels

codecContext->channels = frame->channels;

frame->channels之的是通道的数目。音频一般有双通道或者单通道之分，一般都是双通道吧，我们的程序里面也是设置为双通道的。也就是frame->channels=2.

3.frame->format

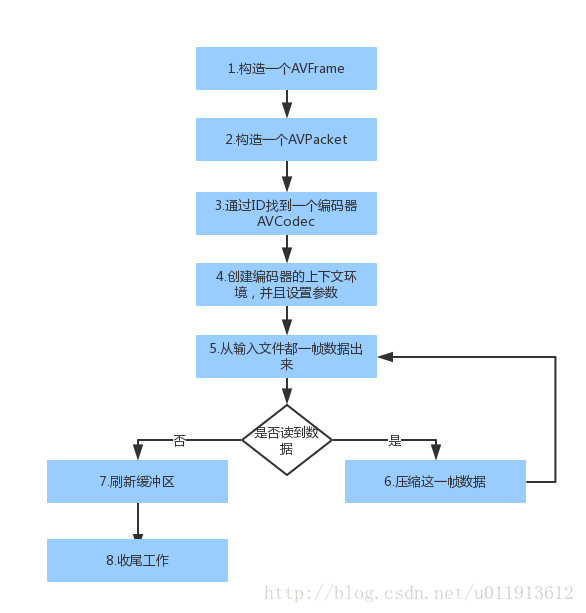
codecContext->sample\_fmt = frame->format;

frame->format指的是样本的格式。一个音频的样本一般用两个字节来描述，分为大小端。我们的程序中使用的是16bit的小端格式。

4.channel\_layout

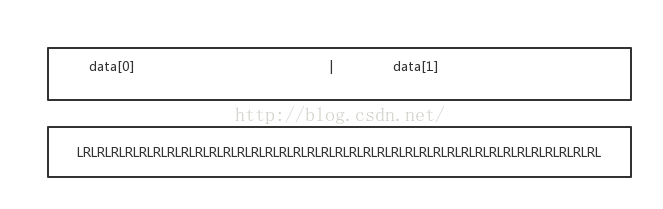
codecContext->channel\_layout = AV\_CH\_LAYOUT\_STEREO;

channel\_layout 用来设置输出通道布局。这个参数不太理解！！！

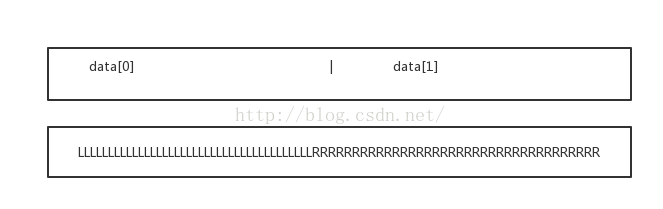


由于AVframe结构体data指针数组不同指针代表指向不同声道的数据，所以产生错误。

上述代码，data指向情况：



而FFmpeg编码PCM为AAC时，需要的是：



所以，我要让到读取一帧时，刚好让data[0]指向一个声道的数据，而data[1]指向另一个声道的数据。

### 源码分析

### 结果验证

编码aac文件

./3-6-encode\_aac ../f32lex2\_44100.pcm test.aac

使用平台的播放播放test.aac即可。

### 参考

1. FFmpeg(2016)PCM编码AAC <http://blog.csdn.net/jammg/article/details/52684894>
2. 将H264和AAC封装为FLV格式

## 将H264和AAC封装为FLV格式

### 框架

### 代码分析

### 结果验证

### 参考

1. ffmpegj将h264与aac封装成mkv/ts或者h264与mp3封装成mp4/flv/avi测试代码http://blog.csdn.net/chenyefei/article/details/52329988

(8)将YUV和PCM编码后封装为FLV格式

(9)将FLV格式文件转封装为MP4/TS/MKV

(10)视频改变缩放比例,并保存为原始格式

(11)音频图像处理（调整亮度、对比度等）

(12)音频重采样

(13)音频音效处理（重低音）

(14)视频转码H264转MP4

(15)视频转码MP4转码H264

(16)音频MP3转码AAC

(17)音频AAC转码MP3

(18)增加水印

(19)为视频添加/更换背景音乐

(20)两段视频做拼接，不同封装不同的解码器，输出指定为FLV封装，H264+AAC编码

(21)实现画中画效果并重新编码。

## 转行格式

一种是annexb模式，传统模式，有startcode，SPS和PPS是在ES中；另一种是mp4模式，一般mp4、mkv、avi会没有startcode，SPS和PPS以及其它信息被封装在container中，每一个frame前面是这个frame的长度，很多解码器只支持annexb这种模式，因此需要将mp4做转换；在ffmpeg中用h264\_mp4toannexb\_filter可以做转换；所以需要使用-bsf h264\_mp4toannexb来进行转换；

# 播放器设计

## 框架

## 所涉及的技术