Android 资料

# 提问问题点:

你是怎么明确每一天的工作的

用于版本控制的工具是什么

在这里工作你喜欢什么

你写单元测试吗

你喜欢单元测试还是集成测试？

你有验收测试吗？

你使用什么测试框架？你喜欢它吗？

您的单元测试要运行多长时间？

**你有持续集成吗？**

好的软件开发团队使用 Jenkins，Travis 和 Buildbot 等工具

代码审查服务Gerrit，GitHub，GitLab，Review Board

10.如何管理bug

如何优先处理错误？

使用什么错误跟踪器？ （和你讨厌什么）

使用Excel来跟踪错误吗？ （nooo！）

bug跟踪器有多少个bug？

错误需要多长时间修复（最小/最大/典型）？

* 作为一个Developer , 不仅要Deliver working code, 还要Deliver maintainable code.
* 必要时进行重构，随着项目的迭代，在计划新增功能的同时，开发要主动计划重构的工作项。
* 开放的心态，虚心接受大家的Review Comments。

**你使用什么框架？**

# ****微信登录内存泄漏****

马伟奇:https://mp.weixin.qq.com/s/GYp2ptku3MaNQHSG7CIfMw

# android8.0新特性:

<https://mp.weixin.qq.com/s/h_0Gk1KPpM2P5KL5HsZdjA>

# 题目锦集

刷题: LeetCode/牛课网

Android客户端面试锦集:

<https://www.jianshu.com/p/cf5092fa2694>

Android阿里面试锦集:

<https://blog.csdn.net/u010054982/article/details/75322180>

# 简历模板:

<http://j.codekk.com/blogs/detail/5705bcdf4a38205862ef476f>

# 问题点:

 1. Android一些优化方案

 2. 什么是过渡绘制，如何防止过渡绘制

 2. 事件分发机制

 3. ListView的优化

 4. Binder机制

 5. 在多进程中，Application会启动几次

 6. 单例模式，双锁原理，volatile原理，静态内部类实现单例的原理

7. Java多线程，synchronized

 8. 聊项目，都具体做了什么。

 9.Binder机制

1. 问了关于数据库的一些问题，SQLite的相关操作，没办法，我在华为唯一一个做的和java相关的项目，但是不太擅长数据库。
2. 网络相关的问题，网络的五层模型，又问了TCP和UDP，还有Android相关的长连接，这里问的比较深。
3. 开始Android先关的知识，Handler机制，Acticity的生命周期以及四种启动模式，各自特点，Service的两种启动模式生命周期，如果同时使用两种启动模式是否可以，顺势讲到Binder，什么时候可以得到控件的大小，最后写了一道链表反转的算法题
4. volley的源代码，在图片缓存部分讨论了挺长时间，http中缓存机制，Last-Modify的作用等。
5. fragment的生命周期
6. service一些知识
7. 事件分发机制
8. Binder实现机制，Stub类中asInterface函数作用，BnBinder和BpBinder区别。
9. gradle中buildToolsVersion和TargetSdkVersion的区别是什么
10. 手机适配一些方案
11. hashmap的实现原理
12. 静态方法是否能被重写
13. 3次握手和4次挥手的原因，以及为什么需要这样做。
14. 数据结构，搜索二叉树的一些特性，平衡二叉树。
15. hashmap是如何解决hash冲突的
16. 进程与线程区别
17. 写了一个二分查找和单例模式
18. http中的同步和异步
19. 聊了一些项目上做的东西,问了问职业规划
20. 我认为Android做的优秀的几个地方，然后又根据我说的问了问比较深入问题。
21. Android是如何进行资源管理的。
22. java比较重要的几个特性
23. 网络五层结构，每一层协议，由于我网络不是很好，还问了一些其他的问题（例如MAC地址和ip地址的区别等）。
24. 为什么离开原来公司，以及职业规划，然后因为面试完大概就晚上8点了，就先让我回去，下周让hr跟我联系，我想这是应该通过面试了吧。
25. OpenGL、GLSurfaceView和Shade
26. 单例，快排，回型打印二维数组
27. 写一个死锁，死锁是怎样产生的，怎样防止死锁
28. Acticity启动模式有几种，分别代表什么意思
29. 聊项目都做了些什么

需要说出时间和空间复杂度，以及有没有更优解。

1. 字符串反转，讨论复杂度。
2. 聊经历和项目
3. 给定一个int型 n，输出1~n的字符串例如 n = 4 输出“1 2 3 4”
4. 输出所有的笛卡尔积组合
5. 单例模式
6. 最长上升子序列
7. 对刚才做的题，一些错的地方做了讨论，主要是类的加载和内部类方面的知识，尤其一些深入到JVM层的知识的确自己有一些不足，不过大部分都还是能答出来。
8. 双亲委托机制，类的五个加载过程。
9. java类加载器的加载流程。
10. Binder机制，从java到framework再到kenral层，面试官问的都很详细，遇到不会的也都会跟我解释。
11. Serializable和Parcelable的区别。
12. View的绘制流程，详细到framework代码，讨论到不确定的地方，立马打开电脑和我一起看源码。
13. 事件分发机制。
14. 常用设计模式，使用静态内部类实现单例的原理是什么。
15. 线程同步，CopyOnWriteArrayList怎样实现。
16. Volley源码，EventBus源码。
17. 项目介绍，聊一些项目上的问题
18. 下拉listview的实现，需要注意的地方和优化的地方（不是lsitview的优化，是下拉的优化）。
19. Android上一些优化方案
20. 介绍了一下现在做的项目，大体说明了现在所用到的技术。
21. 聊项目，GLSufaceView，OpengGL，Shader相关一些知识，在项目中做的优化
22. gc是根据什么来回收一个对象的，引用计数和gc root分别怎么实现。
23. 怎样会产生内存泄露，举一个具体的例子，使用什么检查内存泄露
24. 自定义View的绘制流程
25. 给出二叉树和一个值，找出所有和为这个值的路径；{1,3}{3,6}{3,4}{6,8}区间去重，最少去掉几个集合，可以让这个集合没有交集。
26. 什么是ANR，Activity、BroadcastReceiver、Service对ANR时间限制分别是多少，怎么处理ANR，除了系统生成trace.txt文件，怎么在程序中检测ANR。写出伪代码。
27. 编解码一些知识，I帧，B帧，P帧区别。
28. Android一些优化方案。
29. 算法 n/m，怎么判断得数是无限循环小数
30. 介绍项目。
31. 事件分发机制。
32. 算法，将一个字符串转换成int型数字，考虑 错误输入，溢出，正负值等一些条件，考细心。
33. 给出一个搜索二叉树，输出一个排序好的双向链表。
34. 类的加载过程，Person person = new Person();为例进行说明。
35. JVM相关知识，GC机制。
36. 类的加载器，双亲机制，Android的类加载器。
37. Android中进程的级别，以及各自的区别。
38. 插件化原理。
39. 集合框架，list，map，set都有哪些具体的实现类，区别都是什么。
40. concurrentHashmap原理，原子类。
41. volatile原理。
42. Binder机制。
43. View的绘制流程，事件传递机制，Handler异步消息机制。
44. Interger中的128(-128~127),这里考的是常量池的概念，这个没回答上来。
45. 线程池的相关知识。
46. Handler机制，HandlerThread实现等等。
47. LRUCache算法是怎样实现的。
48. 说了说项目中相关的知识。
49. 聊了聊之前做过的项目。
50. 内存泄露，怎样查找，怎么产生的内存泄露。
51. 怎样计算一张图片的大小，加载bitmap过程（怎样保证不产生内存溢出），二级缓存，LRUCache算法。
52. 简单介绍一下java中的泛型，泛型擦除以及相关的概念。
53. Android优化相关的只是，布局优化、内存优化等等。
54. SQL相关的知识优化的方案，这里我直接说不太了解，就过了。
55. 算法题，m \* n的矩阵，能形成几个正方形（2 \* 2能形成1个正方形，2 \* 3 2个，3 \* 3 6个）
56. Android优化.。
57. 插件化相关技术，热修补技术是怎样实现的，和插件化有什么区别。
58. 是否能讲解清楚你所做的项目，以及使用到的相关知识。
59. Android基础知识
60. Java基础知识，大概是多线程，线程安全，集合类，JVM，类相关知识等。
61. Android一些源码的阅读
62. 优秀的第三方框架源码阅读

* 广度：比较新的技术（Rxjava，插件化等），JVM，GC，网络，数据库，Android（一般不会询问之前面试官问过的问题）。
* 深度：一般会通过1或2个问题来考察，例如：map接口下都有什么子类->hashmap和hashtable区别->hashmap实现原理->怎么解决hash冲突->是否了解concurrentHashmap->concurrentHashmap实现原理->volatile实现原理（concurrentHashmap读是不加锁的，使用到了volatile）
* Bitmap 使用时候注意什么？
* Oom 是否可以try catch ？
* 内存泄露如何产生？
* 适配器模式，装饰者模式，外观模式的异同？
* ANR 如何产生？
* String buffer 与string builder 的区别？
* 如何保证线程安全？
* java四中引用
* Jni 用过么？
* 多进程场景遇见过么？
* 关于handler，在任何地方new handler 都是什么线程下
* sqlite升级，增加字段的语句
* bitmap recycler 相关
* 强引用置为null，会不会被回收？
* glide 使用什么缓存？
* Glide 内存缓存如何控制大小？
* 如何保证多线程读写文件的安全？
* 线程和进程的区别？
* 为什么要有线程，而不是仅仅用进程？
* 算法判断单链表成环与否？
* 如何实现线程同步？
* hashmap数据结构？
* arraylist 与 linkedlist 异同？
* object类的equal 和hashcode 方法重写，为什么？
* hashmap如何put数据（从hashmap源码角度讲解）？
* 简述IPC？
* fragment之间传递数据的方式？
* 简述tcp四次挥手?
* threadlocal原理
* 内存泄漏的可能原因？
* 用IDE如何分析内存泄漏？
* OOM的可能原因？
* 线程死锁的4个条件？
* 差值器&估值器
* 简述消息机制相关
* 进程间通信方式？
* Binder相关？
* 触摸事件的分发？
* 简述Activity启动全部过程？
* okhttp源码？
* RxJava简介及其源码解读？
* 性能优化如何分析systrace？
* 广播的分类？
* 点击事件被拦截，但是相传到下面的view，如何操作？
* Glide源码？
* ActicityThread相关？
* volatile的原理
* synchronize的原理
* lock原理
* 翻转一个单项链表 1->2->3->4->5->null =====> 5->4->3->2->1->null
* string to integer
* 合并多个单有序链表（假设都是递增的）
* Activity生命周期简述
* .常见内存泄漏情景及避免内存泄漏的措施
* Actvity启动模式简述
* 简绘观察者设计模式UML图
* 算法，求公共子序列（或者是子串，记不清了）
* Java四种引用
* 自定义view重写哪几个方法？
* http 的session&cookie的区别
* 简述工作线程更新UI的方法
* 应用最多占多少内存
* 滑动卡顿如何解决（不同原因及对应处理方式）
* 自定义view实战
* 多线程，多进程 相关
* Java四种引用的使用
* XX项目你负责什么
* Sqlite 怎么增加一个字段
* XX项目中是怎么创建数据库的
* Sqlite 怎么删除一个字段
* 有什么你觉得自己做得好的地方
* 为什么用Retrofit（一个开源库）
* Retrofit与之前的网络库有什么优势
* XX项目中你们自己定义的线程池来管理任务，不使用框架，那么，后来新的项目怎么设计的
* 你认为Rxjava的线程池与你们自己实现任务管理框架有什么区别？
* 内存泄漏的常见场景
* 怎么发现&分析内存泄漏
* 处理有序数组为什么比无序数组更快 参考StackOverflow
* 热修复与插件化相关
* Integer类是不是线程安全的，为什么
* 不使用同步锁如何实现线程安全
* 面试头条的时候在线编程：从上到下从左到右输出二叉树
* 针对concurrent包下面的一些类的问题
* Activity面试题
* Fragment面试题
* Service面试题
* Broadcast Receiver面试题
* WebView面试题
* Binder面试题
* Handler面试题
* AsyncTask面试题
* HandlerThread面试题
* IntentService面试题
* 视图工作机制面试题
* 事件分发机制面试题
* ListView面试题
* Android项目构建面试题
* ANR面试题
* OOM面试题
* Bitmap面试题
* UI卡顿面试题
* 内存泄漏面试题
* 内存管理面试题
* 冷启动和热启动面试题
* 其他优化面试题
* 架构模式面试题
* 插件化面试题
* 热更新面试题
* 进程保活面试题
* Lint面试题
* Kotlin面试题

# 阿里热修复:

<https://mp.weixin.qq.com/s/EdMvakAuV5CpUsFl0EvFqw>

# 滴滴插件化:

<https://mp.weixin.qq.com/s/WB0s8oWTX8JeyLyFuRh-ZA>

# 推荐一些值得订阅的 Android 技术专栏

https://mp.weixin.qq.com/s/sVbQ7Yxqn7Mv4isTlZS\_Tw

# 基于项目的问题

1. 你在项目中碰到的最大问题是什么，你是怎么解决的
2. 从这个项目中你学到了什么
3. 什么时候会和其他团队成员有什么样的冲突，你们是怎么解决冲突的？

# 基于开源项目搭建属于自己的技术堆栈

<https://mp.weixin.qq.com/s/vuOiOBOpjlJaaZ6Mp7nPLg>

# 开源项目:

## 基于支付宝首页交互效果：

<https://mp.weixin.qq.com/s/ddTe5gu0orSIM1oG_BDblA>

完整项目包括Rxjava，EvenBus等实现项目:

<https://www.jianshu.com/p/052d69d72c96>

白蓝应用市场

<https://mp.weixin.qq.com/s/DmGMYI_4oMx-EYAcgbDvRA>

https://mp.weixin.qq.com/s?\_\_biz=MzAxMTI4MTkwNQ==&mid=2650824517&idx=1&sn=4336cd916cf069aad01606aad0fce6ef&chksm=80b78bdbb7c002cdcd3adcf7aa493e0d4f1c594bc6806ba91e68fa55244c819c21ddc72fd87f&scene=21#wechat\_redirect

# 书籍:

深入理解Java虚拟机

Android开发艺术探索

疯狂Java讲义

Head First 设计模式

Android群英传

Android开发艺术探索

大话数据结构

面试资料整理:

<https://github.com/LRH1993/android_interview>

<https://legacy.gitbook.com/book/lrh1993/android_interview_guide/details>

<https://blog.csdn.net/lmj623565791>

<http://gityuan.com/>

https://blog.piasy.com/

<http://www.jianshu.com/u/90ab66c248e6>

比较系统的整理相关的类容:

https://github.com/francistao/LearningNotes

<https://juejin.im/post/58a6c38861ff4b0062ae4c25>

<http://www.wanandroid.com/article/list/0?cid=73>

比较普遍的面试大纲

<https://mp.weixin.qq.com/s/bvB2U0-6ZJ1j06iVV4NmjQ>

<https://mp.weixin.qq.com/s/YVvV3-RUjbqYo-DYY3E6nA>

# 音视频:

h.264、AAC、FLV、TS、JPEG分析器

H.264: <https://sourceforge.net/projects/h264streamanalysis/>

AAC：<https://sourceforge.net/projects/aacstreamanalysis/>

FLV: <https://sourceforge.net/projects/flvformatanalysis/>

TS: <http://sourceforge.net/projects/tsformatanalysis/>

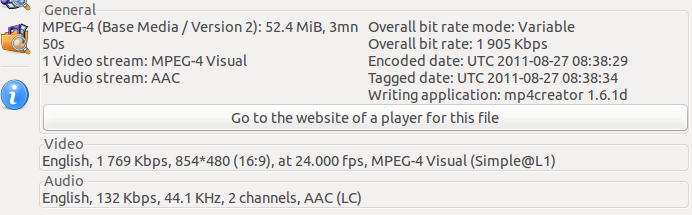
JPEG: <http://sourceforge.net/projects/jpeganalysis/>

音视频编解码完整教程:

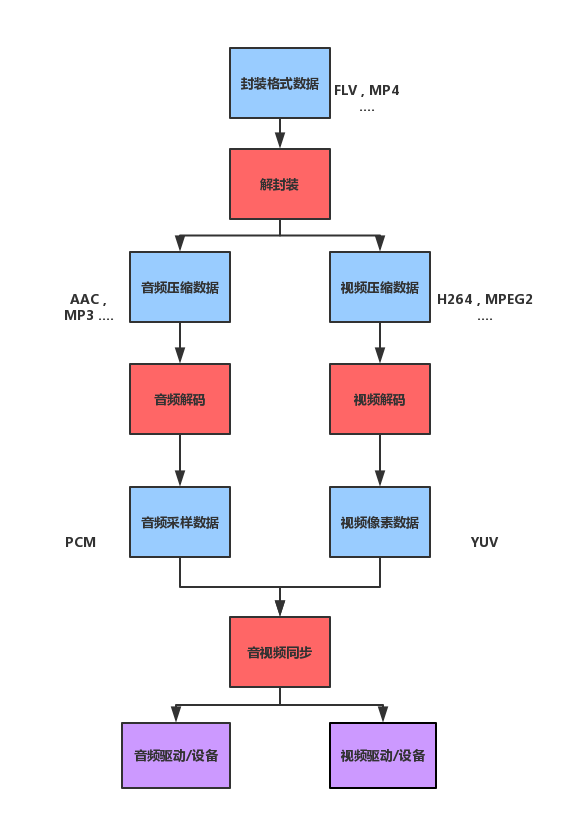
<https://www.cnblogs.com/CoderTian/p/6791638.html>

# 1.播放多媒体文件步骤

通常情况下，我们下载的视频文件如MP4，MKV、FLV等都属于封装格式，就是把音视频数据按照相应的规范，打包成一个文本文件。我们可以使用MediaInfo这个工具查看媒体文件的相关信息。



所以当我们播放一个媒体文件时，通常需要经过以下几个步骤



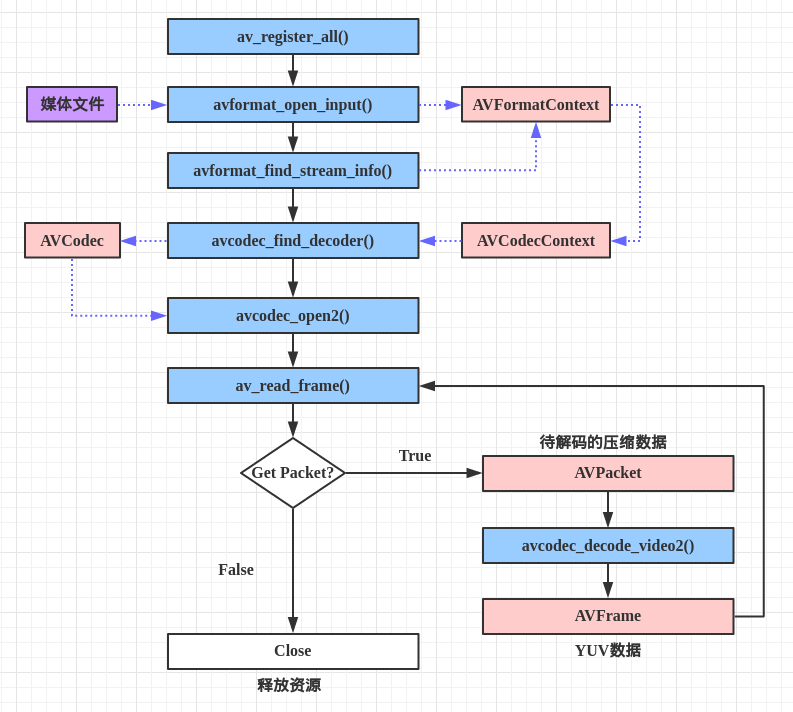
①解封装(Demuxing)：就是将输入的封装格式的数据，分离成为音频流压缩编码数据和视频流压缩编码数据。封装格式种类很多，例如MP4，MKV，RMVB，TS，FLV，AVI等等，它的作用就是将已经压缩编码的视频数据和音频数据按照一定的格式放到一起。例如，FLV格式的数据，经过解封装操作后，输出H.264编码的视频码流和AAC编码的音频码流。

②解码(Decode)：就是将视频/音频压缩编码数据，解码成为非压缩的视频/音频原始数据。音频的压缩编码标准包含AAC，MP3等，视频的压缩编码标准则包含H.264，MPEG2等。解码是整个系统中最重要也是最复杂的一个环节。通过解码，压缩编码的视频数据输出成为非压缩的颜色数据，例如YUV、RGB等等；压缩编码的音频数据输出成为非压缩的音频抽样数据，例如PCM数据。

③音视频同步：就是根据解封装模块处理过程中获取到的参数信息，同步解码出来的音频和视频数据，并将音视频频数据送至系统的显卡和声卡播放出来(Render)。

# 2.FFMPEG音视频解码

通过上面对媒体文件播放步骤的了解，我们在解码多媒体文件的时候需要经过两个步骤，即解封装(Demuxing)和解码(Decode)。下面就来看一下FFMPEG解码媒体文件的时候是怎样做这两个步骤的。



在使用FFMPEG解码媒体文件之前，我们首先需要注册FFMPEG的各种组件，通过

av\_register\_all();

这个函数，可以注册所有支持的容器和对应的codec。之后我们通过

AVFormatContext \*pFormatCtx = avformat\_alloc\_context();

avformat\_open\_input(&pFormatCtx,input\_cstr,NULL,NULL);

avformat\_find\_stream\_info(pFormatCtx,NULL);

来打开一个媒体文件，并获得媒体文件封装格式的上下文。之后我们就可以通过遍历定义在libavformat/avformat.h里保存着媒体文件中封装的流数量的nb\_streams在媒体文件中分离出音视频流。

分离出音视频流之后，就可以对音视频流分别进行解码了，这里先以视频解码为例,我们可以遍历AVStream找到codec\_type为AVMEDIA\_TYPE\_VIDEO的的AVStream即为视频流的索引值。

[制代码](javascript:void(0);)

//视频解码，需要找到视频对应的AVStream所在pFormatCtx->streams的索引位置

int video\_stream\_idx = -1;

int i = 0;

for(; i < pFormatCtx->nb\_streams;i++){

//根据类型判断，是否是视频流

if(pFormatCtx->streams[i]->codec->codec\_type == AVMEDIA\_TYPE\_VIDEO){

video\_stream\_idx = i;

break;

}

}

[制代码](javascript:void(0);)

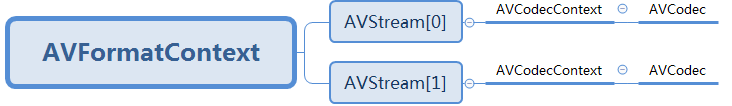
然后我们就可以通过AVStream来找到对应的AVCodecContext即编解码器的上下文。之后就可以通过这个上下文，使用

avcodec\_find\_decoder()

来找到对应的解码器，再通过

avcodec\_open2()

来打开解码器，AVFormatContext、AVStream、AVCodecContext、AVCodec四者之间的关系为



打开解码器之后，就可以循环的将一帧待解码的数据AVPacket送给

avcodec\_decode\_video2()

进行解码，解码之后的数据存放在AVFrame里面。

# 3.示例代码

## 3.1.视频解码

按 Ctrl+C 复制代码

#include <stdio.h>  
#include <stdlib.h>  
//编码  
#include "libavcodec/avcodec.h"  
//封装格式处理  
#include "libavformat/avformat.h"  
//像素处理  
#include "libswscale/swscale.h"  
  
int main()  
{  
  //获取输入输出文件名  
  const char \*input = "test.mp4";  
  const char \*output = "test.yuv";  
  
  //1.注册所有组件  
  av\_register\_all();  
  
  //封装格式上下文，统领全局的结构体，保存了视频文件封装格式的相关信息  
  AVFormatContext \*pFormatCtx = avformat\_alloc\_context();  
  
  //2.打开输入视频文件  
  if (avformat\_open\_input(&pFormatCtx, input, NULL, NULL) != 0)  
  {  
  printf("%s","无法打开输入视频文件");  
  return;  
  }  
  
  //3.获取视频文件信息  
  if (avformat\_find\_stream\_info(pFormatCtx,NULL) < 0)  
  {  
  printf("%s","无法获取视频文件信息");  
  return;  
  }  
  
  //获取视频流的索引位置  
  //遍历所有类型的流（音频流、视频流、字幕流），找到视频流  
  int v\_stream\_idx = -1;  
  int i = 0;  
  //number of streams  
  for (; i < pFormatCtx->nb\_streams; i++)  
  {  
  //流的类型  
  if (pFormatCtx->streams[i]->codec->codec\_type == AVMEDIA\_TYPE\_VIDEO)  
  {  
  v\_stream\_idx = i;  
  break;  
  }  
  }  
  
  if (v\_stream\_idx == -1)  
  {  
  printf("%s","找不到视频流\n");  
  return;  
  }  
  
  //只有知道视频的编码方式，才能够根据编码方式去找到解码器  
  //获取视频流中的编解码上下文  
  AVCodecContext \*pCodecCtx = pFormatCtx->streams[v\_stream\_idx]->codec;  
  //4.根据编解码上下文中的编码id查找对应的解码  
  AVCodec \*pCodec = avcodec\_find\_decoder(pCodecCtx->codec\_id);  
  if (pCodec == NULL)  
  {  
  printf("%s","找不到解码器\n");  
  return;  
  }  
  
  //5.打开解码器  
  if (avcodec\_open2(pCodecCtx,pCodec,NULL)<0)  
  {  
  printf("%s","解码器无法打开\n");  
  return;  
  }  
  
  //输出视频信息  
  printf("视频的文件格式：%s",pFormatCtx->iformat->name);  
  printf("视频时长：%d", (pFormatCtx->duration)/1000000);  
  printf("视频的宽高：%d,%d",pCodecCtx->width,pCodecCtx->height);  
  printf("解码器的名称：%s",pCodec->name);  
  
  //准备读取  
  //AVPacket用于存储一帧一帧的压缩数据（H264）  
  //缓冲区，开辟空间  
  AVPacket \*packet = (AVPacket\*)av\_malloc(sizeof(AVPacket));  
  
  //AVFrame用于存储解码后的像素数据(YUV)  
  //内存分配  
  AVFrame \*pFrame = av\_frame\_alloc();  
  //YUV420  
  AVFrame \*pFrameYUV = av\_frame\_alloc();  
  //只有指定了AVFrame的像素格式、画面大小才能真正分配内存  
  //缓冲区分配内存  
  uint8\_t \*out\_buffer = (uint8\_t \*)av\_malloc(avpicture\_get\_size(AV\_PIX\_FMT\_YUV420P, pCodecCtx->width, pCodecCtx->height));  
  //初始化缓冲区  
  avpicture\_fill((AVPicture \*)pFrameYUV, out\_buffer, AV\_PIX\_FMT\_YUV420P, pCodecCtx->width, pCodecCtx->height);  
  
  //用于转码（缩放）的参数，转之前的宽高，转之后的宽高，格式等  
  struct SwsContext \*sws\_ctx = sws\_getContext(pCodecCtx->width,pCodecCtx->height,pCodecCtx->pix\_fmt,  
  pCodecCtx->width, pCodecCtx->height, AV\_PIX\_FMT\_YUV420P,  
  SWS\_BICUBIC, NULL, NULL, NULL);  
  int got\_picture, ret;  
  
  FILE \*fp\_yuv = fopen(output, "wb+");  
  
  int frame\_count = 0;  
  
  //6.一帧一帧的读取压缩数据  
  while (av\_read\_frame(pFormatCtx, packet) >= 0)  
  {  
  //只要视频压缩数据（根据流的索引位置判断）  
  if (packet->stream\_index == v\_stream\_idx)  
  {  
  //7.解码一帧视频压缩数据，得到视频像素数据  
  ret = avcodec\_decode\_video2(pCodecCtx, pFrame, &got\_picture, packet);  
  if (ret < 0)  
  {  
  printf("%s","解码错误");  
  return;  
  }  
  
  //为0说明解码完成，非0正在解码  
  if (got\_picture)  
  {  
  //AVFrame转为像素格式YUV420，宽高  
  //2 6输入、输出数据  
  //3 7输入、输出画面一行的数据的大小 AVFrame 转换是一行一行转换的  
  //4 输入数据第一列要转码的位置 从0开始  
  //5 输入画面的高度  
  sws\_scale(sws\_ctx, pFrame->data, pFrame->linesize, 0, pCodecCtx->height,  
  pFrameYUV->data, pFrameYUV->linesize);  
  
  //输出到YUV文件  
  //AVFrame像素帧写入文件  
  //data解码后的图像像素数据（音频采样数据）  
  //Y 亮度 UV 色度（压缩了） 人对亮度更加敏感  
  //U V 个数是Y的1/4  
  int y\_size = pCodecCtx->width \* pCodecCtx->height;  
  fwrite(pFrameYUV->data[0], 1, y\_size, fp\_yuv);  
  fwrite(pFrameYUV->data[1], 1, y\_size / 4, fp\_yuv);  
  fwrite(pFrameYUV->data[2], 1, y\_size / 4, fp\_yuv);  
  
  frame\_count++;  
  printf("解码第%d帧\n",frame\_count);  
  }  
  }  
  
  //释放资源  
  av\_free\_packet(packet);  
  }  
  
  fclose(fp\_yuv);  
  
  av\_frame\_free(&pFrame);  
  
  avcodec\_close(pCodecCtx);  
  
  avformat\_free\_context(pFormatCtx);  
  
}

按 Ctrl+C 复制代码

## 3.2.音频解码

[制代码](javascript:void(0);)

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

//封装格式

#include "libavformat/avformat.h"

//解码

#include "libavcodec/avcodec.h"

//缩放

#include "libswscale/swscale.h"

#include "libswresample/swresample.h"

int main (void)

{

//1.注册组件

av\_register\_all();

//封装格式上下文

AVFormatContext \*pFormatCtx = avformat\_alloc\_context();

//2.打开输入音频文件

if (avformat\_open\_input(&pFormatCtx, "test.mp3", NULL, NULL) != 0) {

printf("%s", "打开输入音频文件失败");

return;

}

//3.获取音频信息

if (avformat\_find\_stream\_info(pFormatCtx, NULL) < 0) {

printf("%s", "获取音频信息失败");

return;

}

//音频解码，需要找到对应的AVStream所在的pFormatCtx->streams的索引位置

int audio\_stream\_idx = -1;

int i = 0;

for (; i < pFormatCtx->nb\_streams; i++) {

//根据类型判断是否是音频流

if (pFormatCtx->streams[i]->codec->codec\_type == AVMEDIA\_TYPE\_AUDIO) {

audio\_stream\_idx = i;

break;

}

}

//4.获取解码器

//根据索引拿到对应的流,根据流拿到解码器上下文

AVCodecContext \*pCodeCtx = pFormatCtx->streams[audio\_stream\_idx]->codec;

//再根据上下文拿到编解码id，通过该id拿到解码器

AVCodec \*pCodec = avcodec\_find\_decoder(pCodeCtx->codec\_id);

if (pCodec == NULL) {

printf("%s", "无法解码");

return;

}

//5.打开解码器

if (avcodec\_open2(pCodeCtx, pCodec, NULL) < 0) {

printf("%s", "编码器无法打开");

return;

}

//编码数据

AVPacket \*packet = av\_malloc(sizeof(AVPacket));

//解压缩数据

AVFrame \*frame = av\_frame\_alloc();

//frame->16bit 44100 PCM 统一音频采样格式与采样率

SwrContext \*swrCtx = swr\_alloc();

//重采样设置选项-----------------------------------------------------------start

//输入的采样格式

enum AVSampleFormat in\_sample\_fmt = pCodeCtx->sample\_fmt;

//输出的采样格式 16bit PCM

enum AVSampleFormat out\_sample\_fmt = AV\_SAMPLE\_FMT\_S16;

//输入的采样率

int in\_sample\_rate = pCodeCtx->sample\_rate;

//输出的采样率

int out\_sample\_rate = 44100;

//输入的声道布局

uint64\_t in\_ch\_layout = pCodeCtx->channel\_layout;

//输出的声道布局

uint64\_t out\_ch\_layout = AV\_CH\_LAYOUT\_MONO;

swr\_alloc\_set\_opts(swrCtx, out\_ch\_layout, out\_sample\_fmt, out\_sample\_rate, in\_ch\_layout, in\_sample\_fmt,

in\_sample\_rate, 0, NULL);

swr\_init(swrCtx);

//重采样设置选项-----------------------------------------------------------end

//获取输出的声道个数

int out\_channel\_nb = av\_get\_channel\_layout\_nb\_channels(out\_ch\_layout);

//存储pcm数据

uint8\_t \*out\_buffer = (uint8\_t \*) av\_malloc(2 \* 44100);

FILE \*fp\_pcm = fopen("out.pcm", "wb");

int ret, got\_frame, framecount = 0;

//6.一帧一帧读取压缩的音频数据AVPacket

while (av\_read\_frame(pFormatCtx, packet) >= 0) {

if (packet->stream\_index == audio\_stream\_idx) {

//解码AVPacket->AVFrame

ret = avcodec\_decode\_audio4(pCodeCtx, frame, &got\_frame, packet);

if (ret < 0) {

printf("%s", "解码完成");

}

//非0，正在解码

if (got\_frame) {

printf("解码%d帧", framecount++);

swr\_convert(swrCtx, &out\_buffer, 2 \* 44100, frame->data, frame->nb\_samples);

//获取sample的size

int out\_buffer\_size = av\_samples\_get\_buffer\_size(NULL, out\_channel\_nb, frame->nb\_samples,

out\_sample\_fmt, 1);

//写入文件进行测试

fwrite(out\_buffer, 1, out\_buffer\_size, fp\_pcm);

}

}

av\_free\_packet(packet);

}

fclose(fp\_pcm);

av\_frame\_free(&frame);

av\_free(out\_buffer);

swr\_free(&swrCtx);

avcodec\_close(pCodeCtx);

avformat\_close\_input(&pFormatCtx);

return 0;

}

[制代码](javascript:void(0);)

分类: [FFMPEG学习笔记](http://www.cnblogs.com/CoderTian/category/955003.html)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/708332/20141229093046.png](http://home.cnblogs.com/u/CoderTian/)

[CentTian](http://home.cnblogs.com/u/CoderTian/)  
[关注 - 0](http://home.cnblogs.com/u/CoderTian/followees)  
[粉丝 - 9](http://home.cnblogs.com/u/CoderTian/followers)

[+加关注](javascript:void(0);)

1

0

[«](http://www.cnblogs.com/CoderTian/p/6719277.html)上一篇：[数据结构与算法笔记（3） 排序算法基础](http://www.cnblogs.com/CoderTian/p/6719277.html)  
[»](http://www.cnblogs.com/CoderTian/p/6920306.html)下一篇：[H.264学习笔记](http://www.cnblogs.com/CoderTian/p/6920306.html)