原 [总结]FFMPEG视音频编解码零基础学习方法

置顶　2013年11月16日 00:04:05　阅读数：510149

在CSDN上的这一段日子，接触到了很多同行业的人，尤其是使用FFMPEG进行视音频编解码的人，有的已经是有多年经验的“大神”，有的是刚开始学习的初学者。在和大家探讨的过程中，我忽然发现了一个问题：在“大神”和初学者之间好像有一个不可逾越的鸿沟。“大神”们水平高超，探讨着深奥的问题；而初学者们还停留在入门阶段。究竟是什么原因造成的这种“两极分化”呢？最后，我发现了问题的关键：FFMPEG难度比较大，却没有一个循序渐进，由简单到复杂的教程。现在网上的有关FFMPEG的教程多半难度比较大，不太适合刚接触FFMPEG的人学习；而且很多的例子程序编译通不过，极大地打消了学习的积极性。我自己在刚开始学习FFMPEG的时候也遇到了很大的困难。为了帮助更多的人快速成为“大神”，我想总结一个学习FFMPEG的方法，方便大家循序渐进的学习FFMPEG。

PS：有不少人不清楚“FFmpeg”应该怎么读。它读作“ef ef em peg”

0. 背景知识

本章主要介绍一下FFMPEG都用在了哪里（在这里仅列几个我所知的，其实远比这个多）。说白了就是为了说明：FFMPEG是非常重要的。

使用FFMPEG作为内核视频播放器：

Mplayer，ffplay，射手播放器，暴风影音，KMPlayer，QQ影音...

使用FFMPEG作为内核的Directshow Filter：

ffdshow，lav filters...

使用FFMPEG作为内核的转码工具：

ffmpeg，格式工厂...

事实上，FFMPEG的视音频编解码功能确实太强大了，几乎囊括了现存所有的视音频编码标准，因此只要做视音频开发，几乎离不开它。

对于完全没有视音频技术背景的人来说，在学习FFmpeg之前最好先了解一下几种最基本的视音频数据的格式，可以参考下面的文章：

[总结]视音频编解码技术零基础学习方法

视音频数据处理入门：RGB、YUV像素数据处理

视音频数据处理入门：PCM音频采样数据处理

视音频数据处理入门：H.264视频码流解析

视音频数据处理入门：AAC音频码流解析

视音频数据处理入门：FLV封装格式解析

视音频数据处理入门：UDP-RTP协议解析

1. ffmpeg程序的使用（ffmpeg.exe，ffplay.exe，ffprobe.exe）

【视频资源】

本文中第1,2章是FFmpeg编程最基础的内容。这部分的内容我在给大二同学代课的时候录制成了视频，有时间的话可以看一下《基于 FFmpeg + SDL 的视频播放器的制作》课程的视频。

本章主要介绍一下ffmpeg工程包含的三个exe的使用方法。

ffmpeg的官方网站是： http://ffmpeg.org/

编译好的windows可用版本的下载地址（官网中可以连接到这个网站，和官方网站保持同步）： http://ffmpeg.zeranoe.com/builds/

该网站中的FFMPEG分为3个版本：Static，Shared，Dev。

前两个版本可以直接在命令行中使用，他们的区别在于：Static里面只有3个应用程序：ffmpeg.exe，ffplay.exe，ffprobe.exe，每个exe的体积都很大，相关的Dll已经被编译到exe里面去了。Shared里面除了3个应用程序：ffmpeg.exe，ffplay.exe，ffprobe.exe之外，还有一些Dll，比如说avcodec-54.dll之类的。Shared里面的exe体积很小，他们在运行的时候，到相应的Dll中调用功能。

Dev版本是用于开发的，里面包含了库文件xxx.lib以及头文件xxx.h，这个版本不包含exe文件。

打开系统命令行接面，切换到ffmpeg所在的目录，就可以使用这3个应用程序了。

1.1 ffmpeg.exe

ffmpeg是用于转码的应用程序。

一个简单的转码命令可以这样写：

将input.avi转码成output.ts，并设置视频的码率为640kbps

```plain
1. ffmpeg -i input.avi -b:v 640k output.ts
```

具体的使用方法可以参考： ffmpeg参数中文详细解释

详细的使用说明（英文）： http://ffmpeg.org/ffmpeg.html

1.2 ffplay.exe

ffplay是用于播放的应用程序。

一个简单的播放命令可以这样写：

播放test.avi

```plain
1. ffplay test.avi
```

具体的使用方法可以参考： ffplay的快捷键以及选项

详细的使用说明（英文）： http://ffmpeg.org/ffplay.html

1.3 ffprobe.exe

ffprobe是用于查看文件格式的应用程序。

这个就不多介绍了。

详细的使用说明（英文）： http://ffmpeg.org/ffprobe.html

2. ffmpeg库的使用：视频播放器

本章开始介绍使用ffmpeg的库进行开发。

2.1 ffmpeg库的配置

从 http://ffmpeg.zeranoe.com/builds/ 网站上

1.下载Dev版本，里面包含了ffmpeg的xxx.h头文件以及xxx.lib库文件。

2.下载Shared版本，里面包含了ffmpeg的dll文件。

3.将这两部分文件拷贝到VC工程下面就可以了

注：可能会出现问题，参见： FFMPEG 库移植到 VC 需要的步骤

如果不想自己手动配置，可以下载已经配置好的工程： 最简单的基于FFMPEG+SDL的视频播放器

2.2 最简单的视频播放器

学习文章《 100行代码实现最简单的基于FFMPEG+SDL的视频播放器 》 中的代码，这是ffmpeg做视频播放器最简单的代码了，是我自己精简出来的，已经不能再简化了，每一行都很重要。

原版是基于SDL1.2的视频播放器，后来更新了基于SDL2.0的最简单的视频播放器： 最简单的基于FFMPEG+SDL的视频播放器 ver2（采用SDL2.0）

上述播放器使用libavformat和libavcodec两个类库完成了视频的解码工作。实际上解码工作只需要libavcodec就可以了。因此更新了一个“纯净”的解码器。该解码器只使用libavcodec完成解码工作： 最简单的基于FFmpeg的解码器-纯净版（不包含libavformat）

ffmpeg的函数介绍： ffmpeg函数介绍

注1：播放视频或音频数据的时候会用到SDL。有关SDL可以参考： SDL介绍

SDL参考文档： SDL GUIDE 中文译本

注2：如果想查看解码后的数据，需要用到 YUV播放器： YUV播放器源代码 或 YUV Player Deluxe 都可以

2.3 相关结构体的研究

ffmpeg的结构体之间的关系参考文章： FFMPEG中最关键的结构体之间的关系

结构体中每个变量的分析，参考文章：

FFMPEG结构体分析：AVFrame

FFMPEG结构体分析：AVFormatContext
FFMPEG结构体分析：AVCodecContext
FFMPEG结构体分析：AVIOContext
FFMPEG结构体分析：AVCodec
FFMPEG结构体分析：AVStream
FFMPEG结构体分析：AVPacket

3. ffmpeg库的使用：音频播放器

3.1 最简单的音频播放器

学习文章《 最简单的基于FFMPEG+SDL的音频播放器 》 中的代码，和最简单的视频播放器一样，这是最简单的音频播放器，每一行代码都很重要。

原版是基于SDL1.2的音频播放器，后来更新了一个基于SDL2.0的最简单的音频播放器： 最简单的基于FFMPEG+SDL的音频播放器 ver2（采用SDL2.0）

注：如果想要查看解码后的数据（PCM数据），需要用到Audition。

4. ffmpeg库的使用：一个真正的播放器——ffplay

4.1 真正的播放器

ffplay流程图如文章 《 FFplay源代码分析：整体流程图 》 所示。ffplay代码比较复杂，但是其核心代码和 《 100行代码实现最简单的基于FFMPEG+SDL的视频播放器 》 是一样的。可以两个工程结合着学习。

ffplay代码简介资料： 如何用FFmpeg编写一个简单播放器

ffplay使用说明： ffplay的快捷键以及选项

ffplay已经移植到VC下的工程： ffplay_vc2005 （别人做的，质量很不错）

ffplay移植到MFC下的工程，包含了简单的图形界面和一些控制按钮： ffplay播放器移植VC的工程：ffplay for MFC

上述软件的代码简介： ffplay for mfc 代码备忘

ffplay.c函数结构简单分析： ffplay.c函数结构简单分析（画图）

5. ffmpeg库的使用：编码

5.1 编码

ffmpeg编码我自己研究的不是很多，可以参考文章 ： 使用FFmpeg类库实现YUV视频序列编码为视频

上面那篇文章是用的类库比较旧，新版类库的的使用可以参考下面几篇文章。

图像的编码可以参考： 最简单的基于FFMPEG的图像编码器（YUV编码为JPEG）

音频的编码可以参考： 最简单的基于FFMPEG的音频编码器（PCM编码为AAC）

视频的编码可以参考： 最简单的基于FFMPEG的视频编码器（YUV编码为H.264）

HEVC（H.265）视频编码可以参考： 最简单的基于FFmpeg的视频编码器-更新版（YUV编码为HEVC(H.265)）

上述编码器使用libavformat和libavcodec两个类库完成了视频的编码工作。实际上编码工作只需要libavcodec就可以了。因此更新了一个“纯净”的编码器。该编码器只使用libavcodec完成编码工作： 最简单的基于FFmpeg的编码器-纯净版（不包含libavformat）

5.2 转码

转码实际上是先解码然后编码。

不进行转码，只进行封装格式转换的程序可参考： 最简单的基于FFMPEG的封装格式转换器（无编解码）

转码程序可参考： 最简单的基于FFMPEG的转码程序

比较复杂的转码程序可以参考ffmpeg.c，它移植到MFC下的工程： ffmpeg转码器移植VC的工程：ffmpeg for MFC

ffmpeg.c函数结构简单分析： ffmpeg.c函数结构简单分析（画图）

6. ffmpeg源代码分析

通晓了ffmpeg库的使用以后，可以看一下ffmpeg的源代码。注意ffmpeg的源代码只有在linux下才能编译，在windows下可以使用MinGW进行编译。推荐使用Eclipse查看ffmpeg的源代码。

有一个很完整的ffmpeg源代码的分析文档： ffdoc

FFmpeg的库函数源代码分析文章列表如下：

【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

偏底层的libavcodec的源代码分析文章列表如下：

【解码- libavcodec H.264 解码器】

FFmpeg的H.264解码器源代码简单分析：概述

FFmpeg的H.264解码器源代码简单分析：解析器（Parser）部分

FFmpeg的H.264解码器源代码简单分析：解码器主干部分

FFmpeg的H.264解码器源代码简单分析：熵解码（EntropyDecoding）部分

FFmpeg的H.264解码器源代码简单分析：宏块解码（Decode）部分-帧内宏块（Intra）

FFmpeg的H.264解码器源代码简单分析：宏块解码（Decode）部分-帧间宏块（Inter）

FFmpeg的H.264解码器源代码简单分析：环路滤波（LoopFilter）部分

【解码-libavcodec HEVC 解码器】

FFmpeg的HEVC解码器源代码简单分析：概述

FFmpeg的HEVC解码器源代码简单分析：解析器（Parser）部分

FFmpeg的HEVC解码器源代码简单分析：解码器主干部分

FFmpeg的HEVC解码器源代码简单分析：CTU解码（CTUDecode）部分-PU

FFmpeg的HEVC解码器源代码简单分析：CTU解码（CTU Decode）部分-TU

FFmpeg的HEVC解码器源代码简单分析：环路滤波（LoopFilter）

7.FFmpeg其它几个类库的使用

7.1.libavfilter（加特效）

AVFilter可以给视音频添加各种滤镜效果。有两个例子，一个是给视频添加水印：

最简单的基于FFmpeg的AVfilter例子（水印叠加）

另一个是给YUV数据加特效：

最简单的基于FFmpeg的AVfilter的例子-纯净版

7.2.libavdevice（读设备）

AVDevice可以读取电脑的多媒体设备的数据，或者输出数据到指定的多媒体设备上。

直接使用ffmpeg.exe命令行工具的文章： FFmpeg获取DirectShow设备数据（摄像头，录屏）

编程方面做了2个有关的例子：

读取摄像头： 最简单的基于FFmpeg的AVDevice例子（读取摄像头）

屏幕录制： 最简单的基于FFmpeg的AVDevice例子（屏幕录制）

7.3.libswscale（图像拉伸，像素格式转换）

Swscale类库可以转换像素数据的格式，同时可以拉伸图像的大小。

libswscale的使用示例： 最简单的基于FFmpeg的libswscale的示例（YUV转RGB）

此外，这个示例还附带了一个程序，用于生成测试图片： 最简单的基于FFmpeg的libswscale的示例附件：测试图片生成工具

8.FFmpeg封装格式的处理

使用FFmpeg进行封装格式的处理，主要是通过AVFormat完成。有关封装格式的处理，做了3个例子：

封装格式转换器： 最简单的基于FFMPEG的封装格式转换器（无编解码）

视音频分离器简化版（demuxer-simple）： 最简单的基于FFmpeg的封装格式处理：视音频分离器简化版（demuxer-simple）

视音频分离器（demuxer）： 最简单的基于FFmpeg的封装格式处理：视音频分离器（demuxer）

视音频复用器（muxer）： 最简单的基于FFmpeg的封装格式处理：视音频复用器（muxer）

9.FFmpeg流媒体方面的应用

使用FFmpeg进行流媒体方面的应用，主要是流媒体的发送和接收。

直接使用ffmpeg.exe命令行工具的文章：

FFmpeg发送流媒体的命令（UDP，RTP，RTMP）

编程方面做了一个例子：

基于FFmpeg的推流器： 最简单的基于FFmpeg的推流器（以推送RTMP为例）

10.FFmpeg的其他杂项

使用FFmpeg读写内存（而非文件）的例子：

内存播放器： 最简单的基于FFmpeg的内存读写的例子：内存播放器

内存转码器： 最简单的基于FFmpeg的内存读写的例子：内存转码器

11. ffmpeg在其它平台下的应用

把FFmpeg应用于Android、IOS、Windows Phone的示例程序可以参考：

最简单的基于FFmpeg的移动端例子：Android HelloWorld

最简单的基于FFmpeg的移动端例子：Android 视频解码器

最简单的基于FFmpeg的移动端例子：Android 视频解码器-单个库版

最简单的基于FFmpeg的移动端例子：Android 推流器

最简单的基于FFmpeg的移动端例子：Android 视频转码器

最简单的基于FFmpeg的移动端例子附件：Android 自带播放器

最简单的基于FFmpeg的移动端例子附件：SDL Android HelloWorld

最简单的基于FFmpeg的移动端例子：IOS HelloWorld

最简单的基于FFmpeg的移动端例子：IOS 视频解码器

最简单的基于FFmpeg的移动端例子：IOS 推流器

最简单的基于FFmpeg的移动端例子：IOS 视频转码器

最简单的基于FFmpeg的移动端例子附件：IOS自带播放器

最简单的基于FFmpeg的移动端例子：Windows PhoneHelloWorld

12. ffmpeg相关工程的学习

学习完成ffmpeg，还可以了解一下基于ffmpeg的相关的多媒体开源工程，在这里推荐以下几个：

12.1 ffdshow

ffdshow是基于ffmpeg的解码器类库libavcodec的DirectShow Filter。广泛安装在PC上。

有关ffdshow的源代码分析文章（更新中）：

ffdshow 源代码分析1 ： 整体结构

ffdshow 源代码分析 2： 位图覆盖滤镜（对话框部分Dialog）

ffdshow 源代码分析 3： 位图覆盖滤镜（设置部分Settings）

ffdshow 源代码分析 4： 位图覆盖滤镜（滤镜部分Filter）

ffdshow 源代码分析 5： 位图覆盖滤镜（总结）

ffdshow 源代码分析 6： 对解码器的dll的封装（libavcodec）

ffdshow 源代码分析 7： libavcodec视频解码器类（TvideoCodecLibavcodec）

ffdshow 源代码分析 8： 视频解码器类（TvideoCodecDec）

ffdshow 源代码分析 9： 编解码器有关类的总结

12.2 LAV filters

LAV Filter是基于ffmpeg的解码器类库libavcodec，以及解封装器类库libavformat的DirectShow Filter。广泛安装在PC上。

有关LAV Filter的源代码分析文章：

LAV Filter 源代码分析 1： 总体结构

LAV Filter 源代码分析 2： LAV Splitter

LAV Filter 源代码分析 3： LAV Video （1）

LAV Filter 源代码分析 4： LAV Video （2）

12.3 Mplayer

Mplayer是Linux下使用最广泛的播放器，也有Windows版本的。其中使用了ffmpeg。

有关Mplayer的源代码分析文章：

MPlayer源代码分析

12.4 Media Player Classic - HC

现在广为使用很多播放器都是构建于Media Player Classic - HC的基础之上的。

有关Media Player Classic - HC的源代码分析文章：

Media Player Classic - HC 源代码分析 1：整体结构
Media Player Classic - HC 源代码分析 2：核心类 （CMainFrame）（1）
Media Player Classic - HC 源代码分析 3：核心类 （CMainFrame）（2）
Media Player Classic - HC 源代码分析 4：核心类 （CMainFrame）（3）
Media Player Classic - HC 源代码分析 5：关于对话框 （CAboutDlg）
Media Player Classic - HC 源代码分析 6：MediaInfo选项卡 （CPPageFileMediaInfo）
Media Player Classic - HC 源代码分析 7：详细信息选项卡（CPPageFileInfoDetails）

12.5 XBMC

XBMC是一个优秀的自由和开源的（GPL）媒体中心软件。

有关XBMC源代码分析文章：

XBMC源代码分析 1：整体结构以及编译方法
XBMC源代码分析 2：Addons（皮肤Skin）
XBMC源代码分析 3：核心部分（core）-综述
XBMC源代码分析 4：视频播放器（dvdplayer）-解码器（以ffmpeg为例）
XBMC源代码简析 5：视频播放器（dvdplayer）-解复用器（以ffmpeg为例）
XBMC源代码分析 6：视频播放器（dvdplayer）-文件头（以ffmpeg为例）
XBMC源代码分析 7：视频播放器（dvdplayer）-输入流（以libRTMP为例）

文章标签： ffmpeg 编解码 学习
个人分类： FFMPEG
所属专栏： FFmpeg