原 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）

2014年10月07日 12:54:03 阅读数：20927

==========

最简单的基于FFmpeg的封装格式处理系列文章列表：

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）

最简单的基于FFmpeg的封装格式处理：视音频分离器（demuxer）

最简单的基于FFmpeg的封装格式处理：视音频复用器（muxer）

最简单的基于FFMPEG的封装格式处理：封装格式转换（remuxer）

==========

简介

打算记录一下基于FFmpeg的封装格式处理方面的例子。包括了视音频分离，复用，封装格式转换。有关封转格式转换的例子在之前的文章：《 最简单的基于FFMPEG的封装格式转换器（无编解码） 》中已经有过记录，不再重复。因此计划写3篇文章分别记录视音频的复用器（Muxer）和分离器（Demuxer）。其中视音频分离器（Demuxer）记录2篇：一篇简单的，一篇标准的。简单的版本更适合初学者学习。

本文是第1篇。首先记录一个基于FFmpeg的视音频分离器简单版（Simplest FFmpeg Demuxer Simple）。视音频分离器（Demuxer）即是将封装格式数据（例如MKV）中的视频压缩数据（例如H.264）和音频压缩数据（例如AAC）分离开。如图所示。在这个过程中并不涉及到编码和解码。

本文记录的程序将一个FLV封装的文件（其中视频编码为H.264，音频编码为MP3）分离成为两个文件：一个H.264编码的视频码流文件，一个MP3编码的音频码流文件。
需要注意的是，本文介绍的是一个简单版的视音频分离器（Demuxer）。该分离器的优点是代码十分简单，很好理解。但是缺点是并不适用于一些格式。对于MP3编码的音频是没有问题的。但是在分离MP4/FLV/MKV等一些格式中的AAC编码的码流的时候，得到的AAC码流是不能播放的。原因是存储AAC数据的AVPacket的data字段中的数据是不包含7字节ADTS文件头的“砍头”的数据，是无法直接解码播放的（当然如果在这些数据前面手工加上7字节的ADTS文件头的话，就可以播放了）。

参考文章： 使用FFMPEG类库分离出多媒体文件中的音频码流

分离某些封装格式中的H.264

分离某些封装格式（例如MP4/FLV/MKV等）中的H.264的时候，需要首先写入SPS和PPS，否则会导致分离出来的数据没有SPS、PPS而无法播放。H.264码流的SPS和PPS信息存储在AVCodecContext结构体的extradata中。需要使用ffmpeg中名称为“h264_mp4toannexb”的bitstream filter处理。有两种处理方式：

（1）使用bitstream filter处理每个AVPacket（简单）

把每个AVPacket中的数据（data字段）经过bitstream filter“过滤”一遍。关键函数是av_bitstream_filter_filter()。示例代码如下。

```cpp
AVBitStreamFilterContext* h264bsfc =  av_bitstream_filter_init("h264_mp4toannexb");
while(av_read_frame(ifmt_ctx, &pkt)>=0){
    if(pkt.stream_index==videoindex){
        av_bitstream_filter_filter(h264bsfc, ifmt_ctx->streams[videoindex]->codec, NULL, &pkt.data, &pkt.size, pkt.data, pkt.size, 0);
        fwrite(pkt.data,1,pkt.size,fp_video);
        //...
    }
    av_free_packet(&pkt);
}
av_bitstream_filter_close(h264bsfc);
```

上述代码中，把av_bitstream_filter_filter()的输入数据和输出数据（分别对应第4,5,6,7个参数）都设置成AVPacket的data字段就可以了。

需要注意的是bitstream filter需要初始化和销毁，分别通过函数av_bitstream_filter_init()和av_bitstream_filter_close()。

经过上述代码处理之后，AVPacket中的数据有如下变化：

*每个AVPacket的data添加了H.264的NALU的起始码{0,0,0,1}

*每个IDR帧数据前面添加了SPS和PPS

（2）手工添加SPS，PPS（稍微复杂）

将AVCodecContext的extradata数据经过bitstream filter处理之后得到SPS、PPS，拷贝至每个IDR帧之前。下面代码示例了写入SPS、PPS的过程。

```cpp
FILE *fp=fopen("test.264","ab");
AVCodecContext *pCodecCtx=...
unsigned char *dummy=NULL;
int dummy_len;
AVBitStreamFilterContext* bsfc =  av_bitstream_filter_init("h264_mp4toannexb");
av_bitstream_filter_filter(bsfc, pCodecCtx, NULL, &dummy, &dummy_len, NULL, 0, 0);
fwrite(pCodecCtx->extradata,pCodecCtx-->extradata_size,1,fp);
av_bitstream_filter_close(bsfc);
free(dummy);
```

然后修改AVPacket的data。把前4个字节改为起始码。示例代码如下所示。

```cpp
char nal_start[]={0,0,0,1};
memcpy(packet->data,nal_start,4);
```

经过上述两步也可以得到可以播放的H.264码流，相对于第一种方法来说复杂一些。

参考文章： 使用FFMPEG类库分离出多媒体文件中的H.264码流

当封装格式为MPEG2TS的时候，不存在上述问题。

流程

程序的流程如下图所示。从流程图中可以看出，将每个通过av_read_frame()获得的AVPacket中的数据直接写入文件即可。

简单介绍一下流程中各个重要函数的意义：
avformat_open_input()：打开输入文件。
av_read_frame()：获取一个AVPacket。
fwrite()：根据得到的AVPacket的类型不同，分别写入到不同的文件中。

代码

下面贴上代码：

```cpp
/**
 * 最简单的基于FFmpeg的视音频分离器（简化版）
 * Simplest FFmpeg Demuxer Simple
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序可以将封装格式中的视频码流数据和音频码流数据分离出来。
 * 在该例子中， 将FLV的文件分离得到H.264视频码流文件和MP3
 * 音频码流文件。
 *
 * 注意：
 * 这个是简化版的视音频分离器。与原版的不同在于，没有初始化输出
 * 视频流和音频流的AVFormatContext。而是直接将解码后的得到的
 * AVPacket中的的数据通过fwrite()写入文件。这样做的好处是流程比
 * 较简单。坏处是对一些格式的视音频码流是不适用的，比如说
 * FLV/MP4/MKV等格式中的AAC码流（上述封装格式中的AAC的AVPacket中
 * 的数据缺失了7字节的ADTS文件头）。
 *
 *
 * This software split a media file (in Container such as
 * MKV, FLV, AVI...) to video and audio bitstream.
 * In this example, it demux a FLV file to H.264 bitstream
 * and MP3 bitstream.
 * Note:
 * This is a simple version of "Simplest FFmpeg Demuxer". It is
 * more simple because it doesn't init Output Video/Audio stream's
 * AVFormatContext. It write AVPacket's data to files directly.
 * The advantages of this method is simple. The disadvantages of
```

```
 * this method is it's not suitable for some kind of bitstreams. For
 * example, AAC bitstream in FLV/MP4/MKV Container Format(data in
 * AVPacket lack of 7 bytes of ADTS header).
 *
 */

#include <stdio.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
//Windows
extern "C"
{
#include "libavformat/avformat.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavformat/avformat.h>
#ifdef __cplusplus
};
#endif
#endif


//'1': Use H.264 Bitstream Filter
#define USE_H264BSF 1

int main(int argc, char* argv[])
{
    AVFormatContext *ifmt_ctx = NULL;
    AVPacket pkt;
    int ret, i;
    int videoindex=-1,audioindex=-1;
    const char *in_filename  = "cuc_ieschool.flv";//Input file URL
    const char *out_filename_v = "cuc_ieschool.h264";//Output file URL
    const char *out_filename_a = "cuc_ieschool.mp3";

    av_register_all();
    //Input
    if ((ret = avformat_open_input(&ifmt_ctx, in_filename, 0, 0)) < 0) {
        printf( "Could not open input file.");
        return -1;
    }
    if ((ret = avformat_find_stream_info(ifmt_ctx, 0)) < 0) {
        printf( "Failed to retrieve input stream information");
        return -1;
    }

    videoindex=-1;
    for(i=0; i<ifmt_ctx->nb_streams; i++) {
        if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
        }else if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_AUDIO){
            audioindex=i;
        }
    }
    //Dump Format------------------
    printf("\nInput Video===========================\n");
    av_dump_format(ifmt_ctx, 0, in_filename, 0);
    printf("\n======================================\n");

    FILE *fp_audio=fopen(out_filename_a,"wb+");
    FILE *fp_video=fopen(out_filename_v,"wb+");

    /*
    FIX: H.264 in some container format (FLV, MP4, MKV etc.) need
    "h264_mp4toannexb" bitstream filter (BSF)
      *Add SPS,PPS in front of IDR frame
      *Add start code ("0,0,0,1") in front of NALU
    H.264 in some container (MPEG2TS) don't need this BSF.
    */
#if USE_H264BSF
    AVBitStreamFilterContext* h264bsfc =  av_bitstream_filter_init("h264_mp4toannexb");
#endif

    while(av_read_frame(ifmt_ctx, &pkt)>=0){
        if(pkt.stream_index==videoindex){
#if USE_H264BSF
            av_bitstream_filter_filter(h264bsfc, ifmt_ctx->streams[videoindex]-
>codec, NULL, &pkt.data, &pkt.size, pkt.data, pkt.size, 0);
#endif
            printf("Write Video Packet. size:%d\tpts:%lld\n",pkt.size,pkt.pts);
            fwrite(pkt.data,1,pkt.size,fp_video);
        }else if(pkt.stream_index==audioindex){
            /*
            AAC in some container format (FLV, MP4, MKV etc.) need to add 7 Bytes
```

```
            ADTS Header in front of AVPacket data manually.
            Other Audio Codec (MP3...) works well.
            */
            printf("Write Audio Packet. size:%d\tpts:%lld\n",pkt.size,pkt.pts);
            fwrite(pkt.data,1,pkt.size,fp_audio);
        }
        av_free_packet(&pkt);
    }

#if USE_H264BSF
    av_bitstream_filter_close(h264bsfc);
#endif

    fclose(fp_video);
    fclose(fp_audio);

    avformat_close_input(&ifmt_ctx);

    if (ret < 0 && ret != AVERROR_EOF) {
        printf( "Error occurred.\n");
        return -1;
    }
    return 0;
}
```

结果

输入文件为：
cuc_ieschool.flv：FLV封装格式数据。

输出文件为：
cuc_ieschool.h264：H.264视频码流数据。
cuc_ieschool.mp3：Mp3音频码流数据。

下载

simplest ffmpeg format

项目主页

SourceForge： https://sourceforge.net/projects/simplestffmpegformat/

Github： https://github.com/leixiaohua1020/simplest_ffmpeg_format

开源中国： http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_format

CSDN下载地址：
http://download.csdn.net/detail/leixiaohua1020/8005317

工程中包含4个例子：

simplest_ffmpeg_demuxer_simple：视音频分离器（简化版）。

simplest_ffmpeg_demuxer：视音频分离器。

simplest_ffmpeg_muxer：视音频复用器。

simplest_ffmpeg_remuxer：封装格式转换器。

更新-1.1==========

修复了以下问题：
(1)Release版本下的运行问题
(2)simplest_ffmpeg_muxer封装H.264裸流的时候丢失声音的错误

CSDN下载

http://download.csdn.net/detail/leixiaohua1020/8284309

更新-1.2 (2015.2.13)==========

这次考虑到了跨平台的要求，调整了源代码。经过这次调整之后，源代码可以在以下平台编译通过：

VC++：打开sln文件即可编译，无需配置。

cl.exe：打开compile_cl.bat即可命令行下使用cl.exe进行编译，注意可能需要按照VC的安装路径调整脚本里面的参数。编译命令如下。

```
[plain]
1. ::VS2010 Environment
2. call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
3. ::include
4. @set INCLUDE=include;%INCLUDE%
5. ::lib
6. @set LIB=lib;%LIB%
7. ::compile and link
8. cl simplest_ffmpeg_demuxer_simple.cpp /link avcodec.lib avformat.lib avutil.lib ^
9. avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib /OPT:NOREF
```

MinGW：MinGW命令行下运行compile_mingw.sh即可使用MinGW的g++进行编译。编译命令如下。

```
[plain]
1. g++ simplest_ffmpeg_demuxer_simple.cpp -g -o simplest_ffmpeg_demuxer_simple.exe \
2. -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

GCC：Linux或者MacOS命令行下运行compile_gcc.sh即可使用GCC进行编译。编译命令如下。

```
[plain]
1. gcc simplest_ffmpeg_demuxer_simple.cpp -g -o simplest_ffmpeg_demuxer_simple.out \
2. -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

PS：相关的编译命令已经保存到了工程文件夹中

CSDN下载地址： http://download.csdn.net/detail/leixiaohua1020/8445303
SourceForge上已经更新。

文章标签： ffmpeg 分离 AVPacket demux
个人分类： 我的开源项目 FFMPEG
所属专栏： FFmpeg