原 最简单的基于FFmpeg的内存读写的例子：内存播放器

2014年10月05日 12:15:44　阅读数：24013

==========

最简单的基于FFmpeg的内存读写的例子系列文章列表：

最简单的基于FFmpeg的内存读写的例子：内存播放器

最简单的基于FFmpeg的内存读写的例子：内存转码器

==========

打算记录两个最简单的FFmpeg进行内存读写的例子。之前的所有有关FFmpeg的例子都是对文件进行操作的。例如《　100行代码实现最简单的基于FFMPEG+SDL的视频播放器 》播放的是一个视频的文件。而《　最简单的基于FFMPEG的转码程序 》也是将一个视频文件转换为另一个视频文件。《　最简单的基于FFmpeg的视频编码器（YUV编码为H.264）　》也是最后编码得到一个H.264视频文件。实际上，并不是所有视频的编码，解码都是针对文件进行处理的。有的时候需要的解码的视频数据在一段内存中。例如，通过其他系统送来的视频数据。同样，有的时候编码后的视频数据也未必要保存成一个文件。例如，要求将编码后的视频数据送给其他的系统进行下一步的处理。以上两种情况就要求FFmpeg不仅仅是对文件进行“读，写”操作，而是要对内存进行“读，写”操作。因此打算记录的两个例子就是使用FFmpeg对内存进行读写的例子。

有关FFmpeg读写内存的例子已经在文章《 ffmpeg 从内存中读取数据(或将数据输出到内存) 》中有过叙述，但是一直没有做完整代码的工程。本文记录《最简单的基于FFmpeg内存播放器》。该例子中，首先将文件中的视频数据通过fread()读取到内存中，然后使用FFmpeg播放内存中的数据。

下篇文章计划记录的第二个例子是《最简单的基于FFmpeg内存转码器》。该例子中，首先将文件中的视频数据通过fread()读取到内存中，然后使用FFmpeg读取该数据并进行转码，接着将转码后的数据输出到另一块内存中，最后将该数据通过fwrite()写入成文件。

关于如何从内存中读取数据在这里不再详述，可以参考文章：

ffmpeg 从内存中读取数据(或将数据输出到内存)

关键点

关键点就两个：

1. 初始化自定义的AVIOContext，指定自定义的回调函数。示例代码如下：

```cpp
//AVIOContext中的缓存
unsigned char *aviobuffer=(unsigned char*)av_malloc(32768);
AVIOContext *avio=avio_alloc_context(aviobuffer, 32768,0,NULL,read_buffer,NULL,NULL);
pFormatCtx->pb=avio;

if(avformat_open_input(&pFormatCtx,NULL,NULL,NULL)!=0){
            printf("Couldn't open inputstream.（无法打开输入流）\n");
            return -1;
}
```

上述代码中，自定义了回调函数read_buffer()。在使用avformat_open_input()打开媒体数据的时候，就可以不指定文件的URL了，即其第2个参数为NULL（因为数据不是靠文件读取，而是由read_buffer()提供）

2. 自己写回调函数。示例代码如下：

```cpp
//Callback
int read_buffer(void *opaque, uint8_t *buf, int buf_size){
    if(!feof(fp_open)){
        inttrue_size=fread(buf,1,buf_size,fp_open);
        return true_size;
    }else{
        return -1;
    }
}
```

当系统需要数据的时候，会自动调用该回调函数以获取数据。这个例子为了简单，直接使用fread()读取数据至内存。回调函数需要格外注意它的参数和返回值。

源代码

下面直接贴上程序的源代码：

```cpp
/**
 * 最简单的基于FFmpeg的内存读写例子（内存播放器）
 * Simplest FFmpeg mem Player
 *
 * 雷霄骅
 * 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
 *
 * 本程序实现了对内存中的视频数据的播放。
 * 是最简单的使用FFmpeg读内存的例子。
 *
 * This software play video data in memory (not a file).
 * It's the simplest example to use FFmpeg to read from memory.
 *
 */


#include <stdio.h>

#define __STDC_CONSTANT_MACROS

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

//Output YUV420P
#define OUTPUT_YUV420P 0
FILE *fp_open=NULL;

//Callback
int read_buffer(void *opaque, uint8_t *buf, int buf_size){
    if(!feof(fp_open)){
        int true_size=fread(buf,1,buf_size,fp_open);
        return true_size;
    }else{
        return -1;
    }

}


int main(int argc, char* argv[])
{

    AVFormatContext *pFormatCtx;
    int             i, videoindex;
    AVCodecContext  *pCodecCtx;
    AVCodec         *pCodec;
    char filepath[]="cuc60anniversary_start.mkv";

    av_register_all();
    avformat_network_init();
    pFormatCtx = avformat_alloc_context();

    fp_open=fopen(filepath,"rb+");
    //Init AVIOContext
    unsigned char *aviobuffer=(unsigned char *)av_malloc(32768);
    AVIOContext *avio =avio_alloc_context(aviobuffer, 32768,0,NULL,read_buffer,NULL,NULL);
    pFormatCtx->pb=avio;

    if(avformat_open_input(&pFormatCtx,NULL,NULL,NULL)!=0){
        printf("Couldn't open input stream.\n");
        return -1;
    }
    if(avformat_find_stream_info(pFormatCtx,NULL)<0){
```

```
        printf("Couldn't find stream information.\n");
        return -1;
    }
    videoindex=-1;
    for(i=0; i<pFormatCtx->nb_streams; i++)
        if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
            break;
        }
    if(videoindex==-1){
        printf("Didn't find a video stream.\n");
        return -1;
    }
    pCodecCtx=pFormatCtx->streams[videoindex]->codec;
    pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
    if(pCodec==NULL){
        printf("Codec not found.\n");
        return -1;
    }
    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
        printf("Could not open codec.\n");
        return -1;
    }
    AVFrame *pFrame,*pFrameYUV;
    pFrame=av_frame_alloc();
    pFrameYUV=av_frame_alloc();
    //uint8_t *out_buffer=(uint8_t *)av_malloc(avpicture_get_size(AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height));
    //avpicture_fill((AVPicture *)pFrameYUV, out_buffer, AV_PIX_FMT_YUV420P, pCodecCtx->width, pCodecCtx->height);
    //SDL----------------------------
    if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }

    int screen_w=0,screen_h=0;
    SDL_Surface *screen;
    screen_w = pCodecCtx->width;
    screen_h = pCodecCtx->height;
    screen = SDL_SetVideoMode(screen_w, screen_h, 0,0);

    if(!screen) {
        printf("SDL: could not set video mode - exiting:%s\n",SDL_GetError());
        return -1;
    }
    SDL_Overlay *bmp;
    bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);
    SDL_Rect rect;
    rect.x = 0;
    rect.y = 0;
    rect.w = screen_w;
    rect.h = screen_h;
    //SDL End------------------------
    int ret, got_picture;

    AVPacket *packet=(AVPacket *)av_malloc(sizeof(AVPacket));

#if OUTPUT_YUV420P
    FILE *fp_yuv=fopen("output.yuv","wb+");
#endif
    SDL_WM_SetCaption("Simplest FFmpeg Mem Player",NULL);

    struct SwsContext *img_convert_ctx;
    img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
    //------------------------------
    while(av_read_frame(pFormatCtx, packet)>=0){
        if(packet->stream_index==videoindex){
            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
            if(ret < 0){
                printf("Decode Error.\n");
                return -1;
            }
            if(got_picture){
                SDL_LockYUVOverlay(bmp);
                pFrameYUV->data[0]=bmp->pixels[0];
                pFrameYUV->data[1]=bmp->pixels[2];
                pFrameYUV->data[2]=bmp->pixels[1];
                pFrameYUV->linesize[0]=bmp->pitches[0];
                pFrameYUV->linesize[1]=bmp->pitches[2];
                pFrameYUV->linesize[2]=bmp->pitches[1];
                sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrameYUV->data, pFrameYUV->linesize);
#if OUTPUT_YUV420P
                int y_size=pCodecCtx->width*pCodecCtx->height;
                fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
                fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
                fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
#endif
                SDL_UnlockYUVOverlay(bmp);

                SDL_DisplayYUVOverlay(bmp, &rect);
```

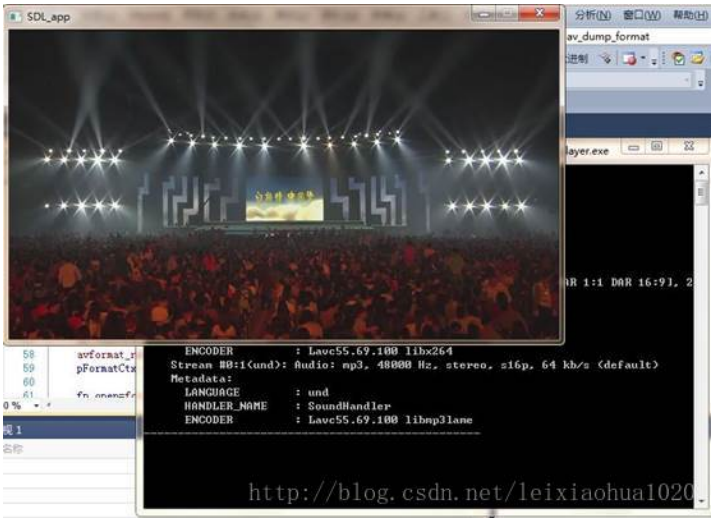
```
                //Delay 40ms
                SDL_Delay(40);
            }
        }
        av_free_packet(packet);
    }
    sws_freeContext(img_convert_ctx);

#if OUTPUT_YUV420P
    fclose(fp_yuv);
#endif

    fclose(fp_open);

    SDL_Quit();

    //av_free(out_buffer);
    av_free(pFrameYUV);
    avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);

    return 0;
}
```

可以通过代码定义的宏来确定是否将解码后的YUV420P数据输出成文件：

```cpp
#define OUTPUT_YUV420P 0
```

结果

程序的运行结果如下。可以解码播放测试视频。适逢60周年校庆，因此截取了一小段校庆晚会的开场画面作为测试视频，给母校庆生~



下载

simplest ffmpeg mem handler

项目主页

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

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

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

CSDN下载地址：
http://download.csdn.net/detail/leixiaohua1020/8003731

本工程包含两个FFmpeg读写内存的例子：
simplest_ffmpeg_mem_player：基于FFmpeg的内存播放器。
simplest_ffmpeg_mem_transcoder：基于FFmpeg的内存转码器（下篇文章记录）。

更新-1.1 (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_mem_player.cpp /MD /link SDL.lib SDLmain.lib avcodec.lib ^
9. avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
10. /SUBSYSTEM:WINDOWS /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_mem_player.cpp -g -o simplest_ffmpeg_mem_player.exe \
2. -I /usr/local/include -L /usr/local/lib \
3. -lmingw32 -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lswscale
```

GCC(Linux)：Linux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_mem_player.cpp -g -o simplest_ffmpeg_mem_player.out -lstdc++ \
2. -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lswscale
```

GCC(MacOS)：Mac终端下运行compile_gcc_mac.sh即可使用Mac 的GCC进行编译，Mac的GCC和Linux的GCC差别不大，但是使用SDL1.2的时候，必须加上"-framework Cocoa"参数，否则编译无法通过。编译命令如下。

```
[plain]
1. gcc simplest_ffmpeg_mem_player.cpp -g -o simplest_ffmpeg_mem_player.out -lstdc++ \
2. -framework Cocoa -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lswscale
```

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/leixiaohua1020/8445795

SourceForge上已经更新。

文章标签： ffmpeg 内存 播放 解码

个人分类： FFMPEG 我的开源项目

所属专栏： FFmpeg