2021年10月

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

公告

昵称：DoubleLi
园龄：11年9个月
粉丝：2080
关注：29
+加关注

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类 (5164)
android(2)
ASP.NET(30)
ASP.NET MVC(11)
Boost(118)
c#(10)
C++/C(778)
c++11(15)
cmake/autotool(66)
com/ATL/Activex(75)
Css(16)
CxImage(12)
darwin stream server(3)
DataBase(32)
DirectX(16)
Extjs(13)
更多

随笔档案 (3864)
2021年10月(33)
2021年9月(4)
2021年8月(10)
2021年7月(43)
2021年6月(1)
2021年5月(29)
2021年4月(15)
2021年3月(13)
2021年2月(96)
2021年1月(47)
2020年12月(2)
2020年11月(27)
2020年10月(44)
2020年9月(14)
2020年8月(4)
更多

文章分类 (2)
SilverLight(1)
sql server(1)

FFmpeg内存操作（三）内存转码器

相关博客列表 ：

FFMPEG内存操作（一） avio_reading.c 回调读取数据到内存解析

FFMPEG内存操作（二）从内存中读取数及数据格式的转换

FFmpeg内存操作（三）内存转码器

本文代码来自于自雷霄骅的《最简单的基于FFmpeg的内存读写的例子：内存转码器》

```
/**
 * This software convert video bitstream (Such as MPEG2) to H.264
 * bitstream. It read video bitstream from memory (not from a file),
 * convert it to H.264 bitstream, and finally output to another memory.
 * It's the simplest example to use FFmpeg to read (or write) from
 * memory.
 *
 */

#include <stdio.h>
#define __STDC_CONSTANT_MACROS

#include "avcodec.h"
#include "avformat.h"
#include "avutil.h"
#include "opt.h"
#include "pixdesc.h"
#include "mathematics.h"

FILEFILE *fp_open;
FILEFILE *fp_write;

//Read File
int read_buffer(voidvoid *opaque, uint8_t *buf, int buf_size){
    int true_size;

    if(!feof(fp_open)){
        true_size=fread(buf,1,buf_size,fp_open);
        printf("read_buffer buf_size = %d\n",buf_size);
        return true_size;
    }else{
        return -1;
    }
}

//Write File
int write_buffer(voidvoid *opaque, uint8_t *buf, int buf_size){
    int true_size;

    if(!feof(fp_write)){
        true_size=fwrite(buf,1,buf_size,fp_write);
        printf("write_buffer buf_size = %d\n",buf_size);
        return true_size;
    }else{
        return -1;
    }
}

```

参考博客
linux驱动
回忆未来-向东
Nginx模块开发与原理剖析
大坡3D软件开发
Dean Chen的专栏
Sloan
音视频FFmpeg等
opencv教程
个人开发历程知识库
关注DirectX
chenyujing1234
morewindows
雷霄骅(leixiaohua1020)的专栏
ffmpeg参考
webrtc参考一
更多

阅读排行榜
1. Nginx之location 匹配规则详解(240898)
2. cmake使用方法详解(178488)
3. MinGW安装和使用(103448)
4. RTMP、RTSP、HTTP视频协议详解（附：直播流地址、播放软件）(102153)
5. C语言字符串操作总结大全(超详细)(94264)

评论排行榜
1. 非IE内核浏览器支持activex插件(37)
2. Nginx之location 匹配规则详解(19)
3. Javascript中定义类(15)
4. C++中的头文件和源文件(9)
5. RTSP协议详解(8)

推荐排行榜
1. C++中的头文件和源文件(25)
2. Nginx之location 匹配规则详解(22)
3. Javascript中定义类(12)
4. JavaScript中typeof知多少？(11)
5. MinGW安装和使用(9)

最新评论
1. Re:windows下搭建nginx-rtmp服务器
configuration-nginx.bat执行报错啊 'auto' 不是内部或外部命令，也不是可运行的程序 或批处理文件。 '--conf-path' 不是内部或外部命令，也不是可运行的程序 或批...
--猫爷°
2. Re:深入理解linux系统下proc文件系统内容

怎么联系作者

--ㅇㅋ
3. Re:go mod模式下引用本地包/模块(module)的方法

go mod用法，不错

--立志做一个好的程序员
4. Re:谷歌浏览器Chrome播放rtsp视频流解决方案
目前市面上已经有很成熟且商用Chrome播放海康威视大华的H.264或H.265的RTSP视频流解决方案了，就是猿大师中间件，底层调用VLC的ActiveX控件可实现在网页中内嵌播放多路RTSP的实时...
--喵大侠

```
int flush_encoder(AVFormatContext *fmt_ctx,unsigned int stream_index)
{
    int ret;
    int got_frame;
    AVPacket enc_pkt;
    if (!(fmt_ctx->streams[stream_index]->codec->codec->capabilities &
                CODEC_CAP_DELAY))
        return 0;
    while (1) {
        av_log(NULL, AV_LOG_INFO, "Flushing stream #%u encoder\n", stream_index);
        //ret = encode_write_frame(NULL, stream_index, &got_frame);
        enc_pkt.data = NULL;
        enc_pkt.size = 0;
        av_init_packet(&enc_pkt);
        ret = avcodec_encode_video2 (fmt_ctx->streams[stream_index]->codec, &enc_pkt,
                NULL, &got_frame);
        av_frame_free(NULL);
        if (ret < 0)
            break;
        if (!got_frame)
        {ret=0;break;}
        /* prepare packet for muxing */
        enc_pkt.stream_index = stream_index;
        enc_pkt.dts = av_rescale_q_rnd(enc_pkt.dts,
                fmt_ctx->streams[stream_index]->codec->time_base,
                fmt_ctx->streams[stream_index]->time_base,
                //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
                (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        enc_pkt.pts = av_rescale_q_rnd(enc_pkt.pts,
                fmt_ctx->streams[stream_index]->codec->time_base,
                fmt_ctx->streams[stream_index]->time_base,
                (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
                //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        enc_pkt.duration = av_rescale_q(enc_pkt.duration,
                fmt_ctx->streams[stream_index]->codec->time_base,
                fmt_ctx->streams[stream_index]->time_base);
        av_log(NULL, AV_LOG_DEBUG, "Muxing frame\n");
        /* mux encoded frame */
        ret = av_write_frame(fmt_ctx, &enc_pkt);
        if (ret < 0)
            break;
    }
    return ret;
}


int main(int argc, char* argv[])
{
    int ret;
    AVFormatContext* ifmt_ctx=NULL;
    AVFormatContext* ofmt_ctx=NULL;
    AVIOContext *avio_in=NULL;
    AVIOContext *avio_out=NULL;
    unsigned char* inbuffer=NULL;
    unsigned char* outbuffer=NULL;
    AVFrame *frame = NULL;
    AVPacket packet;
    AVPacket enc_pkt;

    AVStream *out_stream;
    AVStream *in_stream;
    AVCodecContext *dec_ctx;
    AVCodecContext *enc_ctx;
    AVCodec *encoder;

    AVStream *stream;
    AVCodecContext *codec_ctx;

    enum AVMediaType type;
```

5. Re:如何使用UDP进行跨网段广播
主机A：192.168.3.100 子网掩码 255.255.0.0（手动临时修改）主机B：192.168.120.100 子网掩码 255.255.255.0 主机A 广播 192.168.255...
--zzhilling

```
    unsigned int stream_index;
    unsigned int i=0;
    int got_frame;
    int enc_got_frame;


    fp_open = fopen("cuc60anniversary_start.ts", "rb"); //视频源文件
    fp_write=fopen("cuc60anniversary_start.h264","wb+"); //输出文件

    av_register_all();
    ifmt_ctx=avformat_alloc_context();                                  /* Allocate an AVFormatContext. */
    avformat_alloc_output_context2(&ofmt_ctx, NULL, "h264", NULL);     /*  Allocate an AVFormatContext for an output format. */


    inbuffer=(unsigned char*)av_malloc(32768);
    outbuffer=(unsigned char*)av_malloc(32768);

    /*open input file*/
    avio_in =avio_alloc_context(inbuffer, 32768,0,NULL,read_buffer,NULL,NULL);
    if(avio_in==NULL)
        goto end;
    /*open output file*/
    avio_out =avio_alloc_context(outbuffer, 32768,1,NULL,NULL,write_buffer,NULL);
    if(avio_out==NULL)
        goto end;

    ifmt_ctx->pb=avio_in;                                            /* I/O context.    input output context */
    ifmt_ctx->flags=AVFMT_FLAG_CUSTOM_IO;                            /* The caller has supplied a custom AVIOContext, don't avio_close() it */
    if ((ret = avformat_open_input(&ifmt_ctx, "whatever", NULL, NULL)) < 0) {
        av_log(NULL, AV_LOG_ERROR, "Cannot open input file\n");
        return ret;
    }
    if ((ret = avformat_find_stream_info(ifmt_ctx, NULL)) < 0) {     /*  Read packets of a media file to get stream information */
        av_log(NULL, AV_LOG_ERROR, "Cannot find stream information\n");
        return ret;
    }
    for (i = 0; i < ifmt_ctx->nb_streams; i++) {
        stream = ifmt_ctx->streams[i];
        codec_ctx = stream->codec;
        /* Reencode video & audio and remux subtitles etc. 重新编码视频和音频和翻译字幕等  */
        if (codec_ctx->codec_type == AVMEDIA_TYPE_VIDEO){
            /* Open decoder */
            /* Initialize the AVCodecContext to use the given AVCodec */
            ret = avcodec_open2(codec_ctx,  avcodec_find_decoder(codec_ctx->codec_id), NULL);
            if (ret < 0) {
                av_log(NULL, AV_LOG_ERROR, "Failed to open decoder for stream #%u\n", i);
                return ret;
            }
        }
    }
    //av_dump_format(ifmt_ctx, 0, "whatever", 0);


    //avio_out->write_packet=write_packet;
    ofmt_ctx->pb=avio_out;
    ofmt_ctx->flags=AVFMT_FLAG_CUSTOM_IO;
    for (i = 0; i < 1; i++) {
        out_stream = avformat_new_stream(ofmt_ctx, NULL);    /*  Add a new stream to a media file. */
        if (!out_stream) {
            av_log(NULL, AV_LOG_ERROR, "Failed allocating output stream\n");
            return AVERROR_UNKNOWN;
        }
        in_stream = ifmt_ctx->streams[i];
        dec_ctx = in_stream->codec;
        enc_ctx = out_stream->codec;
        if (dec_ctx->codec_type == AVMEDIA_TYPE_VIDEO)
        {
            encoder = avcodec_find_encoder(AV_CODEC_ID_H264);
            enc_ctx->height = dec_ctx->height;
            enc_ctx->width = dec_ctx->width;
            enc_ctx->sample_aspect_ratio = dec_ctx->sample_aspect_ratio;
            enc_ctx->pix_fmt = encoder->pix_fmts[0];
            enc_ctx->time_base = dec_ctx->time_base;
            //enc_ctx->time_base.num = 1;
            //enc_ctx->time_base.den = 25;
            //H264的必备选项，没有就会错
            enc_ctx->me_range=16;
            enc_ctx->max_qdiff = 4;
            enc_ctx->qmin = 10;
            enc_ctx->qmax = 51;
            enc_ctx->qcompress = 0.6;
            enc_ctx->refs=3;
            enc_ctx->bit_rate = 500000;

            ret = avcodec_open2(enc_ctx, encoder, NULL);
            if (ret < 0) {
                av_log(NULL, AV_LOG_ERROR, "Cannot open video encoder for stream #%u\n", i);
                return ret;
            }
        }
        else if (dec_ctx->codec_type == AVMEDIA_TYPE_UNKNOWN) {
            av_log(NULL, AV_LOG_FATAL, "Elementary stream #%d is of unknown type, cannot proceed\n", i);
            return AVERROR_INVALIDDATA;
        } else {
            /* if this stream must be remuxed */
            /* Copy the settings of the source AVCodecContext into the destination  AVCodecContext */
            ret = avcodec_copy_context(ofmt_ctx->streams[i]->codec,   ifmt_ctx->streams[i]->codec);
            if (ret < 0) {
                av_log(NULL, AV_LOG_ERROR, "Copying stream context failed\n");
                return ret;
            }
        }
        if (ofmt_ctx->oformat->flags & AVFMT_GLOBALHEADER)
            enc_ctx->flags |= CODEC_FLAG_GLOBAL_HEADER;
    }
    //av_dump_format(ofmt_ctx, 0, "whatever", 1);
    /* init muxer, write output file header */

    ret = avformat_write_header(ofmt_ctx, NULL);
    if (ret < 0) {
        av_log(NULL, AV_LOG_ERROR, "Error occurred when opening output file\n");
        return ret;
    }

    i=0;
    /* read all packets */
    while (1) {
        i++;
        if ((ret = av_read_frame(ifmt_ctx, &packet)) < 0)   /* Return the next frame of a stream */
            break;

        stream_index = packet.stream_index;
        if(stream_index!=0)
            continue;

        type = ifmt_ctx->streams[packet.stream_index]->codec->codec_type;
        av_log(NULL, AV_LOG_DEBUG, "Demuxer gave frame of stream_index %u\n", stream_index);
        av_log(NULL, AV_LOG_DEBUG, "Going to reencode the frame\n");

        frame = av_frame_alloc();
        if (!frame) {
            ret = AVERROR(ENOMEM);
            break;
        }

        packet.dts = av_rescale_q_rnd(packet.dts,               /* 解压缩时间戳 */
            ifmt_ctx->streams[stream_index]->time_base,
            ifmt_ctx->streams[stream_index]->codec->time_base,
            //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
            (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        packet.pts = av_rescale_q_rnd(packet.pts,              /* 显示时间戳 */
            ifmt_ctx->streams[stream_index]->time_base,
```

```c
            ifmt_ctx->streams[stream_index]->codec->time_base,
            //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
            (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        /* Decode the video frame of size avpkt->size from avpkt->data into picture 解码输入文件 */
        ret = avcodec_decode_video2(ifmt_ctx->streams[stream_index]->codec, frame, &got_frame, &packet);
        printf("Decode 1 Packet\tsize:%d\tpts:%lld\n",packet.size,packet.pts);

        if (ret < 0) {
            av_frame_free(&frame);
            av_log(NULL, AV_LOG_ERROR, "Decoding failed\n");
            break;
        }
        if (got_frame) {
            frame->pts = av_frame_get_best_effort_timestamp(frame);
            frame->pict_type=AV_PICTURE_TYPE_NONE;

            /* Initialize optional fields of a packet with default values */
            enc_pkt.data = NULL;
            enc_pkt.size = 0;
            av_init_packet(&enc_pkt);

            /* Takes input raw video data from frame and writes the next output packet, if available, to avpkt */
            ret = avcodec_encode_video2 (ofmt_ctx->streams[stream_index]->codec, &enc_pkt, frame, &enc_got_frame);
            printf("Encode 1 Packet\tsize:%d\tpts:%lld\n",enc_pkt.size,enc_pkt.pts);
            av_frame_free(&frame);
            if (ret < 0)
                goto end;
            if (!enc_got_frame)
                continue;

            /* prepare packet for muxing */
            enc_pkt.stream_index = stream_index;
            enc_pkt.dts = av_rescale_q_rnd(enc_pkt.dts,
                ofmt_ctx->streams[stream_index]->codec->time_base,
                ofmt_ctx->streams[stream_index]->time_base,
                //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
                (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
            enc_pkt.pts = av_rescale_q_rnd(enc_pkt.pts,
                ofmt_ctx->streams[stream_index]->codec->time_base,
                ofmt_ctx->streams[stream_index]->time_base,
                //(AVRounding)(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
                (AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
            enc_pkt.duration = av_rescale_q(enc_pkt.duration,
                ofmt_ctx->streams[stream_index]->codec->time_base,
                ofmt_ctx->streams[stream_index]->time_base);
            av_log(NULL, AV_LOG_INFO, "Muxing frame %d\n",i);

            /* mux encoded frame */
            /* Write a packet to an output media file */
            av_write_frame(ofmt_ctx,&enc_pkt);
            if (ret < 0)
                goto end;
        } else {
            av_frame_free(&frame);
        }

        av_free_packet(&packet);
    }

    /* flush encoders */
    for (i = 0; i < 1; i++) {
        /* flush encoder */
        ret = flush_encoder(ofmt_ctx,i);
        if (ret < 0) {
            av_log(NULL, AV_LOG_ERROR, "Flushing encoder failed\n");
            goto end;
        }
    }
    av_write_trailer(ofmt_ctx);
end:
    av_freep(avio_in);
    av_freep(avio_out);
    av_free(inbuffer);
    av_free(outbuffer);
    av_free_packet(&packet);
    av_frame_free(&frame);
    avformat_close_input(&ifmt_ctx);
    avformat_free_context(ofmt_ctx);

    fclose(fp_open);

    if (ret < 0)
        av_log(NULL, AV_LOG_ERROR, "Error occurred\n");
    return (ret? 1:0);
}
```

```
from:http://blog.csdn.net/li_wen01/article/details/64905959
```

分类: ffmpeg、ffplay

好文要顶 关注我 收藏该文

DoubleLi
关注 - 29
粉丝 - 2080
+加关注

0 推荐　0 反对

« 上一篇： 基于v4l2 ffmpeg x264的视频远程监控(附上编译好的库文件)
» 下一篇： FFMPEG内存操作（二）从内存中读取数及数据格式的转换

posted on 2017-08-11 15:38 DoubleLi 阅读(430) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

编辑推荐：
· 聊聊我在微软外服的工作经历及一些个人见解
· 死磕 NIO — Reactor 模式就一定意味着高性能吗？
· 消息队列那么多，为什么建议深入了解下RabbitMQ？
· 技术管理进阶——管人还是管事？
· 以终为始：如何让你的开发符合预期

最新新闻：
· 何小鹏：争取2024年实现飞行汽车量产 价格100万以内（2021-10-24 23:35）
· 供应链危机提振美国在线二手市场 全年销售额预计超650亿美元（2021-10-24 22:00）
· CityTree：一款利用苔藓和机器学习来捕捉空气污染的设备（2021-10-24 20:53）
· 1024程序员节各家怎么过：送霸王洗发水、集体穿格子衫、盲人按摩（2021-10-24 20:00）
· 新卫星图展示泰国季风洪水所带来的巨大影响（2021-10-24 19:00）
» 更多新闻...