

赞

1

收藏

分享

FFmpeg开发常用功能封装

2023-08-24 16:33 - 音视频开发老舅

开发中使用到的FFmpeg常用功能，避免相同功能代码的重复编写，使用时直接复制提升效率。由于音视频处理的场景众多，无法编写完全通用的方法接口，**可能需根据实际场景进行一定的修改，本文章中的代码也将持续更新优化。**

代码

这里提供ffmpegheader.h, ffmpegheader.cpp。配置好基本的FFmpeg库环境后，直接导入上述两个文件，即可直接使用对应功能。

1、头文件

```
#ifndef FFmpegHEADER_H
#define FFmpegHEADER_H
/**
 * 封装常用的ffmpeg方法以及类 只需要引入文件即可直接使用 避免重复轮子
 * By ZXT
 */
extern "C"
{
#include "../libavcodec/avcodec.h"
#include "../libavformat/avformat.h"
#include "../libavformat/avio.h"
#include "../libavutil/opt.h"
#include "../libavutil/time.h"
#include "../libavutil/imgutils.h"
#include "../libswscale/swscale.h"
#include "../libswresample/swresample.h"
#include "../libavutil/avutil.h"
#include "../libavutil/ffversion.h"
#include "../libavutil/frame.h"
#include "../libavutil/pixdesc.h"
#include "../libavutil/imgutils.h"
#include "../libavfilter/avfilter.h"
#include "../libavfilter/buffersink.h"
#include "../libavfilter/buffersrc.h"
#include "../libavdevice/avdevice.h"
}
#include <QDebug>
```

/* ***** 常用函数封装 ***** */

//获取ffmpeg报错信息

```
char *getError(int errNum);
```

//根据pts计算实际时间us

```
int64_t getRealTimeByPTS(int64_t pts, AVRational timebase);
```

//pts转换为us差时进行延时

```
void calcAndDelay(int64_t startTime, int64_t pts, AVRational timebase);
```

//十六进制字节数组转十进制

```
int32_t hexArrayToDec(char *array, int len);
```

/* ***** 常用类封装 ***** */

领音视频学习资料→「链接」

//视频图像转换类

```
class VideoSwser
{
public:
VideoSwser();
~VideoSwser();
```

//初始化转换器

```
bool initSwsCtx(int srcWidth, int srcHeight, AVPixelFormat srcFmt, int dstWidth, int dstHeight, AVPixelFormat dstFmt)
void release();
```

//返回转换格式后的AVFrame

```
AVFrame *getSwsFrame(AVFrame *srcFrame);
private:
```

头条热榜

换一换

开学第一课

- 1 广东全力以赴迎战台风
- 2 荷兰刻机巨头获准年内继续对华出口
- 3 亚运会场馆准备已就绪
- 4 直击台风苏拉轨迹
- 5 深圳停工停业停市
- 6 泰国前总理他信被特赦
- 7 台风“苏拉”是否会影响京津冀
- 8 日本官员为说出“核污水”鞠躬道歉
- 9 深圳全市10级狂风将持续20小时
- 10 国防部回应何时解决台湾问题

这一篇就够了（进阶篇）

206阅读 07月31日

WebRTC 协议推流

190阅读 6天前

查看更多 >

精彩视频

换一换

- 中美高科技战出现重要转折！美小院高墙玩什么... 边打边谈
- 47万次播放
- 我祝你早点消失#日本 #改编歌曲 #音乐 #音乐分享 ... 热歌
- 75万次播放
- 河南鲁山县是个什么地方？管辖它的平顶山市竟然卧虎藏龙
- 49万次播放
- 三小伙回河南老家，阿峰的婚房买到烂尾楼，现... 子整修下
- 29万次播放
- A股成交缩量不到10000亿，明日又将怎么运行？... 把握吗？
- 21万次播放

```
bool hasInit;  
bool needSws;  
int dstWidth;  
int dstHeight;  
AVPixelFormat dstFmt;
```

//格式转换

```
SwsContext *videoSwsCtx;  
};
```

//视频编码器类

```
class VideoEncoder  
{  
public:  
VideoEncoder();  
~VideoEncoder();
```

//初始化编码器

```
bool initEncoder(int width, int height, AVPixelFormat fmt, int fps);  
void release();
```

//返回编码后AVpacket

```
AVPacket *getEncodePacket(AVFrame *srcFrame);  
AVPacket *flushEncoder();
```

//返回编码器上下文

```
AVCodecContext *getCodecContent();  
private:  
bool hasInit;
```

//编码器

```
AVCodecContext *videoEnCodecCtx;  
};
```

//音频重采样类

```
class AudioSwrnr  
{  
public:  
AudioSwrnr();  
~AudioSwrnr();
```

//初始化转换器

```
bool initSwrCtx(int inChannels, int inSampleRate, AVSampleFormat inFmt, int outChar  
void release();
```

//返回转换格式后的AVFrame

```
AVFrame *getSwrFrame(AVFrame *srcFrame);
```

//返回转换格式后的AVFrame srcdata为一帧源格式的数据

```
AVFrame *getSwrFrame(uint8_t *srcData);  
private:  
bool hasInit;  
bool needSwr;  
int outChannels;  
int outSampleRate;  
AVSampleFormat outFmt;
```

//格式转换

```
SwrContext *audioSwrCtx;  
};
```

//音频编码器类

```
class AudioEncoder  
{  
public:  
AudioEncoder();  
~AudioEncoder();
```

//初始化编码器

```
bool initEncoder(int channels, int sampleRate, AVSampleFormat sampleFmt);  
void release();
```

//返回编码后AVpacket

```
AVPacket *getEncodePacket(AVFrame *srcFrame);
AVPacket *flushEncoder();
```

//返回编码器上下文

```
AVCodecContext *getCodecContent();
private:
bool hasInit;
```

//编码器

```
AVCodecContext *audioEnCodecCtx;
};
```

//实时采集场景时间戳处理类

```
class AVTimeStamp
{
public:
```

//累加帧间隔 优点：时间戳稳定均匀 缺点：实际采集帧率可能不稳定，固定累加或忽略小数会累加误差造成不同步

//实时时间戳 优点：时间戳保持实时及正确 缺点：存在帧间隔不均匀，极端情况不能正常播放

//累加+实时矫正 优点：时间戳实时且较为均匀 缺点：纠正时间戳的某一时刻可能画面或声音卡顿

```
enum PTSMode
{
PTS_RECTIFY = 0, //默认矫正类型 保持帧间隔尽量均匀
PTS_REALTIME //实时pts
};
public:
AVTimeStamp();
~AVTimeStamp();
```

//初始化时间戳参数

```
void initAudioTimeStampParam(int sampleRate, PTSMode mode = PTS_RECTIFY);
void initVideoTimeStampParam(int fps, PTSMode mode = PTS_RECTIFY);
```

//开始时间戳记录

```
void startTimeStamp();
```

//返回pts

```
int64_t getAudioPts();
int64_t getVideoPts();
private:
```

//当前模式

```
PTSMode aMode;
PTSMode vMode;
```

//时间戳相关记录 均us单位

```
int64_t startTime;
int64_t audioTimeStamp;
int64_t videoTimeStamp;
double audioDuration;
double videoDuration;
};
#endif // FFMPEGHEADER_H
```

► 2、实现文件

```
#include "ffmpegheader.h"
char *getAVError(int errNum)
{
static char msg[32] = {0};
av_strerror(errNum, msg, 32);
return msg;
}
int64_t getRealTimeByPTS(int64_t pts, AVRational timebase)
{
```

//pts转换为对应us值

```
AVRational timebase_q = {1, AV_TIME_BASE};
int64_t ptsTime = av_rescale_q(pts, timebase, timebase_q);
return ptsTime;
```

```

}
void calcAndDelay(int64_t startTime, int64_t pts, AVRational timebase)
{
    int64_t ptsTime = getRealTimeByPTS(pts, timebase);

```

//计算startTime到此刻的时间差值

```

int64_t nowTime = av_gettime() - startTime;
int64_t offset = ptsTime - nowTime;

```

//大于2秒一般时间戳存在问题 延时无法挽救

```

if(offset > 1000 && offset < 2*1000*1000)
    av_usleep(offset);
}
int32_t hexArrayToDec(char *array, int len)
{

```

//目前限制四字节长度 超过则注意返回类型 防止溢出

```

if(array == nullptr || len > 4)
    return -1;
int32_t result = 0;
for(int i=0; i<len; i++)
    result = result * 256 + (unsigned char)array[i];
return result;
}
VideoSwser::VideoSwser()
{
    videoSwsCtx = nullptr;
    hasInit = false;
    needSws = false;
}
VideoSwser::~VideoSwser()
{
    release();
}
bool VideoSwser::initSwsCtx(int srcWidth, int srcHeight, AVPixelFormat srcFmt, int
{
    release();
    if(srcWidth == dstWidth && srcHeight == dstHeight && srcFmt == dstFmt)
    {
        needSws = false;
    }
    else
    {

```

//设置转换上下文 srcFmt 到 dstFmt(一般为AV_PIX_FMT_YUV420P)的转换

```

videoSwsCtx = sws_getContext(srcWidth, srcHeight, srcFmt, dstWidth, dstHeight, dstF
if (videoSwsCtx == NULL)
{
    qDebug() << "sws_getContext error";
    return false;
}
this->dstFmt = dstFmt;
this->dstWidth = dstWidth;
this->dstHeight = dstHeight;
needSws = true;
}
hasInit = true;
return true;
}
void VideoSwser::release()
{
    if(videoSwsCtx)
    {
        sws_freeContext(videoSwsCtx);
        videoSwsCtx = nullptr;
    }
    hasInit = false;
    needSws = false;
}
AVFrame *VideoSwser::getSwsFrame(AVFrame *srcFrame)
{
    if(!hasInit)
    {
        qDebug() << "Swser未初始化";
        return nullptr;
    }
    if(!srcFrame)
        return nullptr;
    if(!needSws)
        return srcFrame;
    AVFrame *frame = av_frame_alloc();
    frame->format = dstFmt;
    frame->width = dstWidth;
    frame->height = dstHeight;

```

```

int ret = av_frame_get_buffer(frame, 0);
if (ret != 0)
{
    qDebug() << "av_frame_get_buffer swsFrame error";
    return nullptr;
}
ret = av_frame_make_writable(frame);
if (ret != 0)
{
    qDebug() << "av_frame_make_writable swsFrame error";
    return nullptr;
}
sws_scale(videoSwsCtx, (const uint8_t *const *)srcFrame->data, srcFrame->linesize,
return frame;
}
VideoEncoder::VideoEncoder()
{
    videoEnCodecCtx = nullptr;
    hasInit = false;
}
VideoEncoder::~VideoEncoder()
{
    release();
}
bool VideoEncoder::initEncoder(int width, int height, AVPixelFormat fmt, int fps)
{

```

//重置编码信息

```
release();
```

//设置编码器参数 默认AV_CODEC_ID_H264

```

AVCodec *videoEnCoder = avcodec_find_encoder(AV_CODEC_ID_H264);
if(!videoEnCoder)
{
    qDebug() << "avcodec_find_encoder AV_CODEC_ID_H264 error";
    return false;
}
videoEnCodecCtx = avcodec_alloc_context3(videoEnCoder);
if(!videoEnCodecCtx)
{
    qDebug() << "avcodec_alloc_context3 AV_CODEC_ID_H264 error";
    return false;
}

```

//重要！编码参数设置 应根据实际场景修改以下参数

```

videoEnCodecCtx->bit_rate = 2*1024*1024; //1080P:4Mbps 720P:2Mbps 480P:1Mbps 默认中4
videoEnCodecCtx->width = width;
videoEnCodecCtx->height = height;
videoEnCodecCtx->framerate = {fps, 1};
videoEnCodecCtx->time_base = {1, AV_TIME_BASE};
videoEnCodecCtx->gop_size = fps;
videoEnCodecCtx->max_b_frames = 0;
videoEnCodecCtx->pix_fmt = fmt;
videoEnCodecCtx->thread_count = 2;
videoEnCodecCtx->thread_type = FF_THREAD_FRAME;

```

//设置QP最大和最小量化系数，取值范围为0~51 越大编码质量越差

```

videoEnCodecCtx->qmin = 10;
videoEnCodecCtx->qmax = 30;

```

//若设置此项 则sps、pps将保存在extradata; 否则放置于每个帧前

```
videoEnCodecCtx->flags |= AV_CODEC_FLAG_GLOBAL_HEADER;
```

//预设参数 编码速度与压缩率的平衡 如编码快选择的算法就偏简单 压缩率低

//由慢到快veryslow slower slow medium fast faster veryfast superfast ultrafast
默认medium

```

int ret = av_opt_set(videoEnCodecCtx->priv_data, "preset", "ultrafast", 0);
if(ret != 0)
    qDebug() << "av_opt_set preset error";

```

//偏好设置 进行视觉优化

//film电影 animation动画片 grain颗粒物 stillimage静止图片 psnr ssim图像评价指标 fastdecode快速解码 zerolatency零延迟

```

ret = av_opt_set(videoEnCodecCtx->priv_data, "tune", "zerolatency", 0);
if(ret != 0)
    qDebug() << "av_opt_set preset error";

```

//画质设置 可能自动改变 如编码很快很难保证高画质会自动降级

//baseline实时通信 extended使用较少 main流媒体 high广电、存储

```
ret = av_opt_set(videoEncCtx->priv_data, "profile", "main", 0);
if(ret != 0)
qDebug() << "av_opt_set preset error";
ret = avcodec_open2(videoEncCtx, videoEncoder, NULL);
if(ret != 0)
{
qDebug() << "avcodec_open2 video error";
return false;
}
hasInit = true;
return true;
}
void VideoEncoder::release()
{
if(videoEncCtx)
{
avcodec_free_context(&videoEncCtx);
videoEncCtx = nullptr;
}
hasInit = false;
}
AVPacket *VideoEncoder::getEncodePacket(AVFrame *srcFrame)
{
if(!hasInit)
{
qDebug() << "VideoEncoder no init";
return nullptr;
}
if(!srcFrame)
return nullptr;
if(srcFrame->width != videoEncCtx->width
|| srcFrame->height != videoEncCtx->height
|| srcFrame->format != videoEncCtx->pix_fmt)
{
qDebug() << "srcFrame不符合视频编码器设置格式";
return nullptr;
}
}
```

//应保证srcFrame pts为us单位

```
srcFrame->pts = av_rescale_q(srcFrame->pts, AVRational{1, AV_TIME_BASE}, videoEncCtx->time_base);
int ret = avcodec_send_frame(videoEncCtx, srcFrame);
if (ret != 0)
return nullptr;
```



//接收者负责释放packet

```
AVPacket *packet = av_packet_alloc();
ret = avcodec_receive_packet(videoEncCtx, packet);
if (ret != 0)
{
av_packet_free(&packet);
return nullptr;
}
return packet;
}
AVPacket *VideoEncoder::flushEncoder()
{
if(!hasInit)
{
qDebug() << "VideoEncoder no init";
return nullptr;
}
int ret = avcodec_send_frame(videoEncCtx, NULL);
if (ret != 0)
return nullptr;
```

//接收者负责释放packet

```
AVPacket *packet = av_packet_alloc();
ret = avcodec_receive_packet(videoEncCtx, packet);
if (ret != 0)
{
av_packet_free(&packet);
return nullptr;
}
return packet;
}
AVCodecContext *VideoEncoder::getCodecContent()
{
return videoEncCtx;
}
AudioSwrer::AudioSwrer()
{
audioSwrCtx = nullptr;
hasInit = false;
needSwr = false;
```

```

}
AudioSwrnr::~AudioSwrnr()
{
    release();
}
bool AudioSwrnr::initSwrCtx(int inChannels, int inSampleRate, AVSampleFormat inFmt,
{
    release();
    if(inChannels == outChannels && inSampleRate == outSampleRate && inFmt == outFmt)
    {
        needSwr = false;
    }
    else
    {
        audioSwrCtx = swr_alloc_set_opts(NULL, av_get_default_channel_layout(outChannels),
        av_get_default_channel_layout(inChannels), inFmt, inSampleRate, 0, NULL);
        if (!audioSwrCtx)
        {
            qDebug() << "swr_alloc_set_opts failed!";
            return false;
        }
        int ret = swr_init(audioSwrCtx);
        if (ret != 0)
        {
            qDebug() << "swr_init error";
            swr_free(&audioSwrCtx);
            return false;
        }
        this->outFmt = outFmt;
        this->outChannels = outChannels;
        this->outSampleRate = outSampleRate;
        needSwr = true;
    }
    hasInit = true;
    return true;
}
void AudioSwrnr::release()
{
    if(audioSwrCtx)
    {
        swr_free(&audioSwrCtx);
        audioSwrCtx = nullptr;
    }
    hasInit = false;
    needSwr = false;
}
AVFrame *AudioSwrnr::getSwrFrame(AVFrame *srcFrame)
{
    if(!hasInit)
    {
        qDebug() << "Swrnr未初始化";
        return nullptr;
    }
    if(!srcFrame)
    {
        return nullptr;
    }
    if(!needSwr)
    {
        return srcFrame;
    }
    AVFrame *frame = av_frame_alloc();
    frame->format = outFmt;
    frame->channels = outChannels;
    frame->channel_layout = av_get_default_channel_layout(outChannels);
    frame->nb_samples = 1024; //默认aac
    int ret = av_frame_get_buffer(frame, 0);
    if (ret != 0)
    {
        qDebug() << "av_frame_get_buffer audio error";
        return nullptr;
    }
    ret = av_frame_make_writable(frame);
    if (ret != 0)
    {
        qDebug() << "av_frame_make_writable swrFrame error";
        return nullptr;
    }
    const uint8_t **inData = (const uint8_t **)srcFrame->data;
    swr_convert(audioSwrCtx, frame->data, frame->nb_samples, inData, frame->nb_samples);
    return frame;
}
AVFrame *AudioSwrnr::getSwrFrame(uint8_t *srcData)
{
    if(!hasInit)
    {
        qDebug() << "Swrnr未初始化";
        return nullptr;
    }
    if(!srcData)
    {
        return nullptr;
    }
    if(!needSwr)
    {
        return nullptr;
    }
    AVFrame *frame = av_frame_alloc();

```

```

frame->format = outFmt;
frame->channels = outChannels;
frame->sample_rate = outSampleRate;
frame->channel_layout = av_get_default_channel_layout(outChannels);
frame->nb_samples = 1024; //默认aac
int ret = av_frame_get_buffer(frame, 0);
if (ret != 0)
{
    qDebug() << "av_frame_get_buffer audio error";
    return nullptr;
}
ret = av_frame_make_writable(frame);
if (ret != 0)
{
    qDebug() << "av_frame_make_writable swrFrame error";
    return nullptr;
}
const uint8_t *indata[AV_NUM_DATA_POINTERS] = {0};
indata[0] = srcData;
swr_convert(audioSwrCtx, frame->data, frame->nb_samples, indata, frame->nb_samples);
return frame;
}

AudioEncoder::AudioEncoder()
{
    audioEnCodecCtx = nullptr;
    hasInit = false;
}

AudioEncoder::~AudioEncoder()
{
    release();
}

bool AudioEncoder::initEncoder(int channels, int sampleRate, AVSampleFormat sampleF
{
    release();

```

//初始化音频编码器相关 默认AAC

```

AVCodec *audioEnCoder = avcodec_find_encoder(AV_CODEC_ID_AAC);
if (!audioEnCoder)
{
    qDebug() << "avcodec_find_encoder AV_CODEC_ID_AAC failed!";
    return false;
}
audioEnCodecCtx = avcodec_alloc_context3(audioEnCoder);
if (!audioEnCodecCtx)
{
    qDebug() << "avcodec_alloc_context3 AV_CODEC_ID_AAC failed!";
    return false;
}

```

//ffmpeg -h encoder=aac 自带编码器仅支持AV_SAMPLE_FMT_FLTP 大多数AAC 编码器都采用平面布局格式 提高数据访问效率和缓存命中率 加快编码效率

//音频数据量偏小 设置较为简单

```

audioEnCodecCtx->bit_rate = 64*1024;
audioEnCodecCtx->time_base = AVRational{1, sampleRate};
audioEnCodecCtx->sample_rate = sampleRate;
audioEnCodecCtx->sample_fmt = sampleFmt;
audioEnCodecCtx->channels = channels;
audioEnCodecCtx->channel_layout = av_get_default_channel_layout(channels);
audioEnCodecCtx->frame_size = 1024;

```

//打开音频编码器

```

int ret = avcodec_open2(audioEnCodecCtx, audioEnCoder, NULL);
if (ret != 0)
{
    qDebug() << "avcodec_open2 audio error" << getAVError(ret);
    return false;
}
hasInit = true;
return true;
}

void AudioEncoder::release()
{
    if(audioEnCodecCtx)
    {
        avcodec_free_context(&audioEnCodecCtx);
        audioEnCodecCtx = nullptr;
    }
    hasInit = false;
}

AVPacket *AudioEncoder::getEncodePacket(AVFrame *srcFrame)
{
    if(!hasInit)
    {
        qDebug() << "AudioEncoder no init";
        return nullptr;
    }

```



```

    }
    if(!srcFrame)
        return nullptr;
    if(srcFrame->channels != audioEnCodecCtx->channels
    || srcFrame->sample_rate != audioEnCodecCtx->sample_rate
    || srcFrame->format != audioEnCodecCtx->sample_fmt)
    {
        qDebug() << "srcFrame不符合音频编码器设置格式";
        return nullptr;
    }
}

```

//应保证srcFrame pts为us单位

```
srcFrame->pts = av_rescale_q(srcFrame->pts, AVRational{1, AV_TIME_BASE}, audioEnCod
```

//进行音频编码得到编码数据AVPacket

```

int ret = avcodec_send_frame(audioEnCodecCtx, srcFrame);
if (ret != 0)
    return nullptr;

```

//接收者负责释放packet

```

AVPacket *packet = av_packet_alloc();
ret = avcodec_receive_packet(audioEnCodecCtx, packet);
if (ret != 0)
{
    av_packet_free(&packet);
    return nullptr;
}
return packet;
}

AVPacket *AudioEncoder::flushEncoder()
{
    if(!hasInit)
    {
        qDebug() << "AudioEncoder no init";
        return nullptr;
    }
    int ret = avcodec_send_frame(audioEnCodecCtx, NULL);
    if (ret != 0)
        return nullptr;
}

```

//接收者负责释放packet

```

AVPacket *packet = av_packet_alloc();
ret = avcodec_receive_packet(audioEnCodecCtx, packet);
if (ret != 0)
{
    av_packet_free(&packet);
    return nullptr;
}
return packet;
}

AVCodecContext *AudioEncoder::getCodecContent()
{
    return audioEnCodecCtx;
}

AVTimeStamp::AVTimeStamp()
{
    aMode = PTS_RECTIFY;
    vMode = PTS_RECTIFY;
    startTime = 0;
    audioTimeStamp = 0;
    videoTimeStamp = 0;
}

```

//默认视频264编码 25帧

```
videoDuration = 1000000 / 25;
```

//默认音频aac编码 44100采样率

```

audioDuration = 1000000 / (44100 / 1024);
}
AVTimeStamp::~AVTimeStamp()
{
}

void AVTimeStamp::initAudioTimeStampParm(int sampleRate, AVTimeStamp::PTSMODE mode)
{
    aMode = mode;
    audioDuration = 1000000 / (sampleRate / 1024);
}

void AVTimeStamp::initVideoTimeStampParm(int fps, AVTimeStamp::PTSMODE mode)
{
    vMode = mode;
    videoDuration = 1000000 / fps;
}

void AVTimeStamp::startTimeStamp()

```

```
{
    audioTimeStamp = 0;
    videoTimeStamp = 0;
    startTime = av_gettime();
}

int64_t AVTimeStamp::getAudioPts()
{
    if(aMode == PTS_RECTIFY)
    {
        int64_t elapsed = av_gettime() - startTime;
        uint32_t offset = qAbs(elapsed - (audioTimeStamp + audioDuration));
        if(offset < (audioDuration * 0.5))
            audioTimeStamp += audioDuration;
        else
            audioTimeStamp = elapsed;
    }
    else
    {
        audioTimeStamp = av_gettime() - startTime;
    }
    return audioTimeStamp;
}

int64_t AVTimeStamp::getVideoPts()
{
    if(vMode == PTS_RECTIFY)
    {
        int64_t elapsed = av_gettime() - startTime;
        uint32_t offset = qAbs(elapsed - (videoTimeStamp + videoDuration));
        if(offset < (videoDuration * 0.5))
            videoTimeStamp += videoDuration;
        else
            videoTimeStamp = elapsed;
    }
    else
    {
        videoTimeStamp = av_gettime() - startTime;
    }
    return videoTimeStamp;
}
}
```

① 举报

评论 1



聪颖的元气豆张FY

赞

转发了

回复 · 8天前

×

分享一个用python带着原有格式拆分Excel表格到多张工作表的思路



睡觉偷懶的肥猫 08月01日

×

国防部回应何时解决台湾问题

环球网 2.2万评论 20小时前

×



2160公里！杭州父子三人暑期骑行西藏，爸爸瘦16斤，老大瘦10斤



长江日报 2126评论 6天前

×

成都老小区决定自拆自建，专家：第一个吃螃蟹的能提供借鉴意义

齐鲁壹点 1473评论 4天前

×



江苏：落实降低购买首套房首付比例、个人首套房“认房不认贷”等政策

澎湃新闻 627评论 21小时前

×



“如果有来世，我还是愿意嫁给你爸爸”——纪念我的母亲王文娟



新民晚报 779评论 08月06日

×

拿地即开工！总投资31亿元，长城汽车智能核心部件无锡基地项目动工



上观新闻 2小时前



中国知识产权两大盛会“设博会”“专交会”将分别在无锡大连举行

北青网 20小时前



江苏男子腿断了要求送货上门被快递员怼“看看哪条断了”？申通回应

潇湘晨报 5评论 2小时前



爸爸给孩子取名“宇凡”，护士手误多加一笔，妈妈大喜：不改了



小雪妈妈育儿 347评论 6天前



国家终于出手了！5个综艺节目被强制停播，没一个值得同情！



晓瑜海棠 16评论 2小时前



热门：无限流：阴展怪 | 人在原神，开局 | 团宠娇娇女：被 | 隐藏在霍格沃茨 | 少女风水师女团 | 洪荒之度厄逍遥 | 宇宙能源科技有 | 陶里春风三期 | 才子男装官方翻 | 三国之称孤道尊

