原 最简单的基于FFMPEG的推流器附件：收流器

2015年07月18日 08:47:14　阅读数：17655

==========

最简单的基于FFmpeg的推流器系列文章列表：

《最简单的基于FFmpeg的推流器（以推送RTMP为例）》

《最简单的基于FFMPEG的推流器附件：收流器》

==========

出于对《最简单的基于FFmpeg的推流器》的补充，本文记录一个最简单的基于FFmpeg的收流器。收流器和推流器的作用正好相反：推流器用于将本地文件以流媒体的形式发送出去，而收流器用于将流媒体内容保存为本地文件。

本文记录的推流器可以将RTMP流媒体保存成为一个本地的FLV文件。由于FFmpeg本身支持很多的流媒体协议和封装格式，所以也支持其它的封装格式和流媒体协议。

源代码

```cpp
/**
 * 最简单的基于FFmpeg的收流器（接收RTMP）
 * Simplest FFmpeg Receiver (Receive RTMP)
 *
 * 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
 *
 * 本例子将流媒体数据（以RTMP为例）保存成本地文件。
 * 是使用FFmpeg进行流媒体接收最简单的教程。
 *
 * This example saves streaming media data (Use RTMP as example)
 * as a local file.
 * It's the simplest FFmpeg stream receiver.
 *
 */

#include <stdio.h>

#define __STDC_CONSTANT_MACROS

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

//'1': Use H.264 Bitstream Filter
#define USE_H264BSF 0

int main(int argc, char* argv[])
{
    AVOutputFormat *ofmt = NULL;
    //Input AVFormatContext and Output AVFormatContext
```

```
    AVFormatContext *ifmt_ctx = NULL, *ofmt_ctx = NULL;
    AVPacket pkt;
    const char *in_filename, *out_filename;
    int ret, i;
    int videoindex=-1;
    int frame_index=0;
    in_filename  = "rtmp://live.hkstv.hk.lxdns.com/live/hks";
    //in_filename  = "rtp://233.233.233.233:6666";
    //out_filename = "receive.ts";
    //out_filename = "receive.mkv";
    out_filename = "receive.flv";

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

    for(i=0; i<ifmt_ctx->nb_streams; i++)
        if(ifmt_ctx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
            videoindex=i;
            break;
        }

    av_dump_format(ifmt_ctx, 0, in_filename, 0);

    //Output
    avformat_alloc_output_context2(&ofmt_ctx, NULL, NULL, out_filename); //RTMP

    if (!ofmt_ctx) {
        printf( "Could not create output context\n");
        ret = AVERROR_UNKNOWN;
        goto end;
    }
    ofmt = ofmt_ctx->oformat;
    for (i = 0; i < ifmt_ctx->nb_streams; i++) {
        //Create output AVStream according to input AVStream
        AVStream *in_stream = ifmt_ctx->streams[i];
        AVStream *out_stream = avformat_new_stream(ofmt_ctx, in_stream->codec->codec);
        if (!out_stream) {
            printf( "Failed allocating output stream\n");
            ret = AVERROR_UNKNOWN;
            goto end;
        }
        //Copy the settings of AVCodecContext
        ret = avcodec_copy_context(out_stream->codec, in_stream->codec);
        if (ret < 0) {
            printf( "Failed to copy context from input to output stream codec context\n");
            goto end;
        }
        out_stream->codec->codec_tag = 0;
        if (ofmt_ctx->oformat->flags & AVFMT_GLOBALHEADER)
            out_stream->codec->flags |= CODEC_FLAG_GLOBAL_HEADER;
    }
    //Dump Format------------------
    av_dump_format(ofmt_ctx, 0, out_filename, 1);
    //Open output URL
    if (!(ofmt->flags & AVFMT_NOFILE)) {
        ret = avio_open(&ofmt_ctx->pb, out_filename, AVIO_FLAG_WRITE);
        if (ret < 0) {
            printf( "Could not open output URL '%s'", out_filename);
            goto end;
        }
    }
    //Write file header
    ret = avformat_write_header(ofmt_ctx, NULL);
    if (ret < 0) {
        printf( "Error occurred when opening output URL\n");
        goto end;
    }

#if USE_H264BSF
    AVBitStreamFilterContext* h264bsfc =  av_bitstream_filter_init("h264_mp4toannexb");
#endif

    while (1) {
        AVStream *in_stream, *out_stream;
        //Get an AVPacket
        ret = av_read_frame(ifmt_ctx, &pkt);
        if (ret < 0)
            break;

        in_stream  = ifmt_ctx->streams[pkt.stream_index];
        out_stream = ofmt_ctx->streams[pkt.stream_index];
        /* copy packet */
```

```
        /* copy packet */
        //Convert PTS/DTS
        pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream->time_base, (AVRounding)
(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream->time_base, (AVRounding)
(AV_ROUND_NEAR_INF|AV_ROUND_PASS_MINMAX));
        pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
        pkt.pos = -1;
        //Print to Screen
        if(pkt.stream_index==videoindex){
            printf("Receive %8d video frames from input URL\n",frame_index);
            frame_index++;

#if USE_H264BSF
            av_bitstream_filter_filter(h264bsfc, in_stream->codec, NULL, &pkt.data, &pkt.size, pkt.data, pkt.size, 0);
#endif
        }
        //ret = av_write_frame(ofmt_ctx, &pkt);
        ret = av_interleaved_write_frame(ofmt_ctx, &pkt);

        if (ret < 0) {
            printf( "Error muxing packet\n");
            break;
        }

        av_free_packet(&pkt);

    }

#if USE_H264BSF
    av_bitstream_filter_close(h264bsfc);
#endif

    //Write file trailer
    av_write_trailer(ofmt_ctx);
end:
    avformat_close_input(&ifmt_ctx);
    /* close output */
    if (ofmt_ctx && !(ofmt->flags & AVFMT_NOFILE))
        avio_close(ofmt_ctx->pb);
    avformat_free_context(ofmt_ctx);
    if (ret < 0 && ret != AVERROR_EOF) {
        printf( "Error occurred.\n");
        return -1;
    }
    return 0;
}
```

运行结果

程序运行之后，即可获取流媒体数据并且在本地保存成一个视频文件。

下载

simplest ffmpeg streamer

项目主页

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

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

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

CSDN下载地址： http://download.csdn.net/detail/leixiaohua1020/8924345

解决方案包含2个项目：

simplest_ffmpeg_streamer: 将本地视频文件推送至流媒体服务器。

simplest_ffmpeg_receiver: 将流媒体数据保存成本地文件。

文章标签： FFmpeg 流媒体 视频 RTMP

个人分类： FFMPEG

所属专栏： FFmpeg