ajaxhe的专栏

:■ 目录视图

₩ 摘要视图

RSS 订阅

个人资料



ajaxhe



访问: 81960次

积分: 1459分

排名: 第9376名

原创: 54篇 转载: 15篇 译文: 1篇 评论: 98条

文章搜索

文章分类

C/C++ (13)

Gstreamer (2)

UNIX/Linux (24)

WEB (10)

其他 (4)

嵌入式(embed) (5)

matlab (1)

cache优化 (2)

DSP (2)

音频编解码 (9)

Qt (8)

编程之美 (1)

算法 (2)

opencv (2)

文章存档

2014年03月 (2)

2013年11月 (2)

2013年07月 (3)

2013年05月 (1)

2013年03月 (5)

展开

有奖征资源,博文分享有内涵

5月推荐博文汇总

大数据读书汇--获奖名单公布

2014 CSDN博文大赛

ortp学习笔记

分类: C/C++ 音频编解码

2012-07-16 22:07

2885人阅读

评论(9) 收藏 举报

session

codec parameters header

ortp版本: ortp-0.18.0.tar.gz 操作系统: window 7 32bit

video

1.windows下编译ortp.lib

直接打开ortp-0.18.0\build\win32native的工程文件即可,VS2008下无需任何修改,即可编译出动态链接库 ortp.lib以及ortp.dll。

2.使用ortp提供的测试程序: ortp-0.18.0\src\tests下的win_receive以及win_sender目录下程序

在windows7下使用win_receiver时会提示下如下错误,这个在错误在windows XP下是不存在的

QOSAddSocketToFlow failed to add a flow with error 87

具体问题暂时还没有深究,这个应该是一个系统兼容性问题,需要系统支持qwave.lib。

在查找了错误出处,对比了前几个ortp的版本后,对ortp0.18的源代码进行了修改

rtp_session_inet.c

/* set socket options (but don't change chosen states) */

rtp_session_set_dscp(session, -1);

rtp_session_set_multicast_ttl(session, -1);

rtp_session_set_multicast_loopback(session, -1);

3. 发送H.264视频帧

ortp提供的win_receive以及win_sender一次只发送160字节的数据。但我们在发送一帧H.264视频帧,每次需要发送2000-3000字节的 数据,关于ortp发送H.264视频帧,可以参考:

ortp编程示例代码

谈谈RTP传输中的负载类型和时间戳

除了上面两篇文章提到需要注意的负载类型和时间戳之外,在ortp中win_receive中还需要显示调用:

rtp_session_set_recv_buf_size(rtp_session_mgr .rtp_session , recv_bufsize);

这里的recv_bufsize必须要比win_sender中

sended_bytes = rtp_session_send_with_ts (rtp_session_mgr .rtp_session ,

(uint8 t*) send buffer.

wrapLen,

rtp_session_mgr.cur_timestamp);

的wrapLen要大,否则在receive端会出现如下错误:

ortp-warning-Error receiving RTP packet: Error code: 10040, err num [10040], error [-1]

4. 一个结合H.264编码,rtp发送,接收,保存mp4文件的小程序

下载地址: ortp测试程序(修改版)

win7下ffmpeg编译动态粒(5899) 通过ffmpeg将aac格式转(4836) fedora 14 无线网卡(RTL (4607) 修改output-example,将(4299) matlab安装过程错误及解(3851) Qt Phonon+QGraphicsV(2933) ortp学习笔记(2885) Qt+mplayer播放器(2556) AAC Advanced Audio C(2552) Qt:在TreeModel+QTreeV(2489)

评论排行 fedora 14 无线网卡(RTL 修改output-example,将 (13)ortp学习笔记 (9) 通过ffmpeg将aac格式转 (6) AAC Advanced Audio C (6) Ot Phonon+OGraphicsV (5) win7下ffmpeg编译动态链 (4) QSqlite使用整理 (4) Linux (fedora) 菜鸟之抗 (4) PL0语言单词的词法分析 (4)

推荐文章

最新评论

Qt+DirectDraw实现

固本培元: 楼主这篇文章对我们现在一个项目很有帮助,谢谢~

Qt Phonon+QGraphicsView视频 ajaxhe: @volvetandanny:我之前 也考虑过你提到了这种思路,但 一直没有成功。后面就干脆在底 层SDK...

Qt Phonon+QGraphicsView视频 volvetandanny: 请教一个问题 呀,我播放是用自己的SDK来做 的,就是把videowidget的winld传 给底层SDK....

ffmpeg 重写tutorial01程序--将一outblack: // Save the frame to disk if (++i <= 10) 这个地方为...

修改output-example,将H.264,/ zhangk569:楼主,新改过的放在 linux里面交叉编译,还是通不 过。自己写了个Makefile也不 行,望指点。

修改output-example,将H.264,/zhangk569:这个太好了,我先试试

win7下ffmpeg编译动态链接库整: ajaxhe: @xyzw7777:很久以前的 事情,已经忘记当时的情况了。 windows下建议直接使用http://...

win7下ffmpeg编译动态链接库整: xyzw7777: 楼主请问怎么将 ffmpeg编译成1个dll跟1个lib呢?

ortp学习笔记

mengfanxinqq: 你好,这个 QOSAddSocketToFlow failed to add a flow with ...

ortp学习笔记

x152qian: 34是H263,怎么可能会发送成功?网上还多都写34.真不知道发贴的人是否测试过? oRtp里定义了H...

```
编译环境: vs2008
```

压缩文件中包括了ffmpeg以及ortp的动态链接库,但你还需要在工程文件中修改它们的路径才能正确链接这两个库. 因为这个小程序是一个项目的一部分,源代码中还有些冗余代码,并没有来的及删掉,大家在看的时候需要注意下

receiver

```
[cpp]
01.
      #include <string.h>
02.
      #include "ortp/ortp.h"
03.
04.
      extern "C"{
05.
      #include <libavformat/avformat.h>
06.
      #include <libswscale/swscale.h>
07.
      };
08.
      bool m bExit = FALSE:
09.
10.
      struct RtpSessionMar
11.
12.
          RtpSession *rtp_session;
13.
14.
          int timestamp;
15.
      };
16.
17.
18.
      RtpSessionMgr rtp_session_mgr;
      const int timestamp_inc = 3600; // 90000/25
19.
20.
      const char recv_ip[] = "127.0.0.1";
21.
22.
      const int recv_port = 8008;
23.
      const int recv_bufsize = 10240;
24.
      unsigned char *recv_buf;
25.
       ** 帧包头的标识长度 */
26.
27.
      #define CMD_HEADER_LEN 10
28.
29.
      /** 帧包头的定义 */
      static uint8_t CMD_HEADER_STR[CMD_HEADER_LEN] = { 0xAA,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7
30.
31.
        ** 帧的包头信息 */
32.
33.
      typedef struct _sFrameHeader
34.
          /** 命令名称标识 */
35.
          unsigned \  \, \textbf{char} \  \, \textbf{cmdHeader[CMD\_HEADER\_LEN];}
36.
37.
          /** 采集的通道号 0~7*/
38.
39.
          unsigned char chId;
40.
41.
          /** 数据类型, 音频 或者 视频*/
42.
          unsigned char dataType;
43.
44.
          /** 缓冲区的数据长度 */
45.
          uint32 t len;
46.
47.
          /** 时间戳 */
48.
          uint32_t timestamp;
49.
50.
      }FrameHeader;
51.
52.
      53.
      AVOutputFormat *fmt:
54.
      AVFormatContext *oc;
      AVStream *video st:
55.
      AVCodecContext *codecContext;
56.
57.
58.
      const int image_width = 704;
59.
      const int image_height = 576;
60.
      const int frame_rate = 25;
61.
      static int frame_count;
62.
63.
      BOOL ctrlHandlerFunction(DWORD fdwCtrlType)
64.
65.
          switch (fdwCtrlType)
66.
          {
67.
              // Handle the CTRL+C signal.
              // CTRL+CLOSE: confirm that the user wants to exit.
68.
69.
          case CTRL_C_EVENT:
```

```
70.
           case CTRL_CLOSE_EVENT:
 71.
           case CTRL BREAK EVENT:
 72.
           case CTRL_LOGOFF_EVENT:
 73.
           case CTRL SHUTDOWN EVENT:
 74.
               m_bExit = TRUE;
 75.
                return TRUE;
 76.
 77.
           default:
 78.
                return FALSE;
 79.
 80.
       }
 81.
 82.
       void rtpInit()
 83.
 84.
           int ret;
 85.
           WSADATA wsaData;
 86.
 87.
           /** 初始化winsocket */
           if ( WSAStartup(MAKEWORD(2,2), &wsaData) != 0)
 88.
 89.
 90.
                fprintf(stderr, "WAStartup failed!\n");
 91.
 92.
           }
 93.
 94.
           ortp_init();
 95.
           ortp_scheduler_init();
 96.
 97.
           rtp_session_mgr.rtp_session = rtp_session_new(RTP_SESSION_RECVONLY);
 98.
 99.
           rtp_session_set_scheduling_mode(rtp_session_mgr.rtp_session, 1);
100.
           rtp_session_set_blocking_mode(rtp_session_mgr.rtp_session, 1);
101.
           rtp_session_set_local_addr(rtp_session_mgr.rtp_session, recv_ip, recv_port);
102.
103.
           rtp_session_enable_adaptive_jitter_compensation(rtp_session_mgr.rtp_session, TRUE);
104.
           rtp_session_set_jitter_compensation(rtp_session_mgr.rtp_session, 40);
105.
106.
           rtp_session_set_payload_type(rtp_session_mgr.rtp_session, 34);
107.
           rtp_session_set_recv_buf_size(rtp_session_mgr.rtp_session, recv_bufsize);
108.
109.
           rtp session mgr.timestamp = timestamp inc;
110.
       }
111.
112.
       int rtp2disk()
113.
       {
114.
           int err;
115.
           int havemore = 1;
116.
117.
           while (havemore)
118.
119.
                err = rtp_session_recv_with_ts(rtp_session_mgr.rtp_session,
120.
                                                 (uint8_t *)recv_buf, recv_bufsize,
121.
                                                 rtp_session_mgr.timestamp, &havemore);
122.
123.
                if (havemore)
                    printf("==> Warning: havemore=1!\n");
124.
125.
126.
                if (err > 0)
127.
                    FrameHeader *frameHeader;
128.
129.
                    printf("receive data is %d\n", err);
130.
131.
                    frameHeader = (FrameHeader *)recv_buf;
132.
133.
                    printf("frame_len = %d\n", frameHeader->len);
134.
                    AVPacket pkt;
135.
                    av_init_packet(&pkt);
136.
137.
                    pkt.stream_index= video_st->index;
                    pkt.data= recv_buf + sizeof(FrameHeader);
138.
139.
                    pkt.size = frameHeader->len; // not the video_outbuf_size, note!
140.
                    \ensuremath{//} write the compressed frame in the media file
141.
                    err = av_write_frame(oc, &pkt);
142.
143.
                    if (err != 0)
144.
                    {
145.
                        printf("av_write_frame failed\n");
146.
                    }
147.
               }
148.
```

```
149.
           return 0:
150.
151.
       }
152.
153.
       AVCodecContext* createCodecContext(AVFormatContext *oc)
154.
155.
            AVCodecContext *video_cc = avcodec_alloc_context();
156.
157.
            video_cc = avcodec_alloc_context();
158.
            if (!video_cc)
159.
                fprintf(stderr, "alloc avcodec context failed\n");
160.
161.
                exit(1);
162.
           }
163.
164.
            video_cc->codec_id = (CodecID)CODEC_ID_H264;
            video_cc->codec_type = AVMEDIA_TYPE_VIDEO;
165.
166.
167.
           video cc->me range = 16:
168.
            video_cc->max_qdiff = 4;
169.
            video_cc->qmin = 10;
170.
            video_cc->qmax = 51;
171.
           video_cc->qcompress = 0.6f;
172.
173.
            /* put sample parameters */
174.
            video_cc->bit_rate = 400000;
175.
176.
            /* resolution must be a multiple of two */
177.
           video_cc->width = image_width;
178.
           video_cc->height = image_height;
179.
180.
            /* time base: this is the fundamental unit of time (in seconds) in terms
181.
            of which frame timestamps are represented. for fixed-fps content,
182.
            timebase should be 1/framerate and timestamp increments should be
183.
            identically 1. */
            video_cc->time_base.den = frame_rate;
184.
185.
            video_cc->time_base.num = 1;
186.
187.
            video_cc->gop_size = 12; /* emit one intra frame every twelve frames at most */
188.
           video cc->pix fmt = PIX FMT YUV420P;
189.
190.
            // some formats want stream headers to be separate
191.
           if(!strcmp(oc->oformat->name, "mp4") || !strcmp(oc->oformat-
       >name, "mov") || !strcmp(oc->oformat->name, "3gp"))
192.
               video_cc->flags |= CODEC_FLAG_GLOBAL_HEADER;
193.
194.
            return video_cc;
       }
195.
196.
197.
       void openVideo(AVFormatContext *oc)
198.
            AVCodec *codec;
199.
200.
201.
            /* find the video encoder */
202.
           codec = avcodec find encoder(codecContext->codec id);
203.
            if (!codec) {
204.
                fprintf(stderr, "codec not found\n");
205.
                exit(1);
206.
           }
207.
            /* open the codec */
208.
209.
            if (avcodec_open(codecContext, codec) < 0) {</pre>
                fprintf(stderr, "could not open video codec\n");
210.
                exit(1);
211.
212.
           }
       }
213.
214.
215.
       void ffmpegEncodeInit()
216.
            // initialize libavcodec, and register all codecs and formats
217.
218.
           av_register_all();
219.
220.
            char filename[] = "test.mp4";
221.
            fmt = av_guess_format(NULL, filename, NULL);
222.
223.
           oc = avformat_alloc_context();
224.
            oc->oformat = fmt;
225.
226.
           fmt->video_codec = (CodecID) CODEC_ID_H264;
```

ortp学习笔记 - ajaxhe的专栏 - 博客频道 - CSDN.NET

```
227.
           _snprintf(oc->filename, sizeof(oc->filename), "%s", filename);
228.
229.
           // add the video streams using the default format codecs and initialize the codecs
230.
           video_st = NULL;
           if (fmt->video_codec != CODEC_ID_NONE) {
231.
232.
                video_st = av_new_stream(oc, 0);
233.
                if (!video_st) {
                    fprintf(stderr, "Could not alloc stream\n");
234.
235.
                    exit(1);
236.
                }
237.
           }
238.
239.
           // alloc codecContext
240.
           codecContext = createCodecContext(oc);
241.
           video_st->codec = codecContext;
242.
           // set the output parameters (must be done even if no parameters).
243.
244.
           if (av_set_parameters(oc, NULL) < 0) {</pre>
                fprintf(stderr, "Invalid output format parameters\n");
245.
246.
                exit(1);
247.
           }
248.
249.
           dump_format(oc, 0, filename, 1);
250.
           /^{\star} now that all the parameters are set, we can open the audio and
251.
252.
           video codecs and allocate the necessary encode buffers */
253.
           if (codecContext)
254.
                openVideo(oc);
255.
256.
           // open the output file, if needed
257.
           if (!(fmt->flags & AVFMT_NOFILE)) {
258.
                if (url_fopen(&oc->pb, filename, URL_WRONLY) < 0) {</pre>
259.
                    fprintf(stderr, "Could not open '%s'\n", filename);
260.
                    exit(1);
261.
                }
262.
           }
263.
264.
           // write the stream header, if any
265.
           av_write_header(oc);
266.
       }
267.
       void ffmpegEncodeClose()
268.
269.
       {
           int i:
270.
271.
           /* close each codec */
272.
273.
274.
                avcodec_close(video_st->codec);
275.
276.
           // write the trailer, if any
277.
           av_write_trailer(oc);
278.
279.
           /* free the streams */
280.
           for(i = 0; i < oc->nb_streams; i++) {
281.
                av_freep(&oc->streams[i]->codec);
282.
                av_freep(&oc->streams[i]);
283.
           }
284.
285.
           if (!(fmt->flags & AVFMT_NOFILE)) {
286.
                /* close the output file */
287.
                url_fclose(oc->pb);
288.
           }
289.
            /* free the stream */
290.
291.
           av free(oc);
292.
       }
293.
294.
       int main()
295.
       {
           recv_buf = (uint8_t *)malloc(recv_bufsize);
296.
297.
298.
           rtpInit();
299.
           ffmpegEncodeInit();
300.
301.
           // ====== INSTALL THE CONTROL HANDLER =======
302.
           if (SetConsoleCtrlHandler( (PHANDLER_ROUTINE) ctrlHandlerFunction, TRUE) == 0)
303.
304.
                printf("==> Cannot handle the CTRL-C...\n");
305.
```

```
306.
            printf("==> RTP Receiver started\n");
307.
 308.
309.
            while (m_bExit == FALSE)
 310.
            {
311.
                rtp2disk();
312.
313.
                rtp_session_mgr.timestamp += timestamp_inc;
314.
            }
315.
            printf("==> Exiting\n");
316.
317.
318.
            free(recv_buf);
319.
320.
            ffmpegEncodeClose();
321.
322.
            rtp_session_destroy(rtp_session_mgr.rtp_session);
323.
        }
324.
4
sender
        [cpp]
 01.
        #include <ortp/ortp.h>
 02.
        #include <string.h>
 03.
 04.
        extern "C"{
 05.
        #include <libayformat/avformat.h>
        #include <libswscale/swscale.h>
 06.
 07.
        };
 08.
 09.
        struct RtpSessionMgr
 10.
 11.
            RtpSession *rtp_session;
 12.
            uint32_t timestamp_inc;
 13.
            uint32_t cur_timestamp;
 14.
        };
 15.
 16.
        RtpSessionMgr rtp_session_mgr;
 17.
        const char g_ip[] = "127.0.0.1";
 18.
 19.
        const int g_port = 8008;
        const uint32_t timestamp_inc = 3600; // 90000 / 25
 20.
  21.
 22.
        const int image_width = 704;
 23.
        const int image_height = 576;
 24.
        const int frame_rate = 25;
 25.
        static int frame_count, wrap_size;
 26.
 27.
       AVCodecContext *video_cc;
 28.
 29.
        AVFrame *picture;
 30.
        /** 帧包头的标识长度 */
 31.
 32.
        #define CMD_HEADER_LEN 10
 33.
        /** 帧包头的定义 */
 34.
        static uint8_t CMD_HEADER_STR[CMD_HEADER_LEN] = { 0xAA, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7
 35.
 36.
 37.
        /** 帧的包头信息 */
 38.
        typedef struct _sFrameHeader
 39.
 40.
            /** 命令名称标识 */
            unsigned char cmdHeader[CMD_HEADER_LEN];
 41.
 42.
            /** 采集的通道号 0~7*/
 43.
  44.
            unsigned char chId;
 45.
 46.
            /** 数据类型, 音频 或者 视频*/
            unsigned char dataType:
 47.
 48.
            /** 缓冲区的数据长度 */
 49.
 50.
            uint32_t len;
 51.
            /** 时间戳 */
```

```
53.
            uint32_t timestamp;
 54.
 55.
       }FrameHeader;
 56.
 57.
        // set frame header
 58.
       FrameHeader frameHeader;
 59.
 60.
       void rtpInit()
 61.
       {
 62.
            char *m_SSRC;
 63.
 64.
           ortp_init();
 65.
           ortp_scheduler_init();
 66.
            printf("Scheduler initialized\n");
 67.
 68.
            rtp_session_mgr.rtp_session = rtp_session_new(RTP_SESSION_SENDONLY);
 69.
 70.
            rtp_session_set_scheduling_mode(rtp_session_mgr.rtp_session, 1);
           rtp session set blocking mode(rtp session mgr.rtp session, 1);
 71.
 72.
            rtp_session_set_remote_addr(rtp_session_mgr.rtp_session, g_ip, g_port);
 73.
           rtp_session_set_send_payload_type(rtp_session_mgr.rtp_session, 34); // 34 is for H.26
 74.
           m_SSRC = getenv("SSRC");
 75.
 76.
           if (m_SSRC != NULL)
 77.
           {
 78.
                rtp_session_set_ssrc(rtp_session_mgr.rtp_session, atoi(m_SSRC));
 79.
           }
 80.
 81.
           rtp_session_mgr.cur_timestamp = 0;
 82.
            rtp_session_mgr.timestamp_inc = timestamp_inc;
 83.
 84.
           printf("rtp init success!\n");
 85.
       }
 86.
 87.
       int rtpSend(unsigned char *send_buffer, int frame_len)
 88.
       {
 89.
            FrameHeader *fHeader = (FrameHeader *)send_buffer;
 90.
            fHeader->chId
                             = 0;
            fHeader->dataType = 0; // SESSION_TYPE_VIDEO
 91.
                            = frame len;
 92.
            fHeader->len
 93.
            fHeader->timestamp = 0;
 94.
 95.
            printf("frame header len = %d\n", fHeader->len);
 96.
 97.
            int wrapLen;
           wrapLen = frame_len + sizeof(FrameHeader);
 98.
 99.
100.
            int sended bytes:
            sended_bytes = rtp_session_send_with_ts(rtp_session_mgr.rtp_session,
101.
102.
                                                     (uint8 t *)send buffer,
103.
                                                    wrapLen,
104.
                                                     rtp_session_mgr.cur_timestamp);
105.
106.
            rtp_session_mgr.cur_timestamp += rtp_session_mgr.timestamp_inc;
107.
108.
            return sended_bytes;
109.
       }
110.
111.
       void createCodecContext()
112.
       {
           video_cc = avcodec_alloc_context();
113.
114.
           if (!video_cc)
115.
116.
                fprintf(stderr, "alloc avcodec context failed\n");
117.
                exit(1);
118.
           }
119.
120.
            video_cc->codec_id = (CodecID)CODEC_ID_H264;
121.
           video_cc->codec_type = AVMEDIA_TYPE_VIDEO;
122.
123.
           video_cc->me_range = 16;
124.
            video_cc->max_qdiff = 4;
125.
           video_cc->qmin = 10;
126.
           video_cc->qmax = 51;
127.
           video_cc->qcompress = 0.6f;
128.
129.
            /* put sample parameters */
130.
           video_cc->bit_rate = 400000;
131.
```

```
132.
            /* resolution must be a multiple of two */
           video cc->width = image width:
133.
134.
           video_cc->height = image_height;
135.
136.
           /st time base: this is the fundamental unit of time (in seconds) in terms
137.
           of which frame timestamps are represented. for fixed-fps content,
138.
           timebase should be 1/framerate and timestamp increments should be
139.
           identically 1. */
140.
           video_cc->time_base.den = frame_rate;
141.
           video_cc->time_base.num = 1;
142.
143.
           video_cc->gop_size = 12; /* emit one intra frame every twelve frames at most */
144.
           video_cc->pix_fmt = PIX_FMT_YUV420P;
145.
       }
146.
147.
       AVFrame *allocPicture(int pix_fmt, int width, int height)
148.
149.
           AVFrame *picture;
           uint8_t *picture_buf;
150.
151.
           int size;
152.
153.
           picture = avcodec_alloc_frame();
154.
           if (!picture)
155.
156.
157.
           size = avpicture_get_size((PixelFormat)pix_fmt, width, height);
158.
           picture_buf = (uint8_t *)av_malloc(size);
159.
           if (!picture_buf) {
160.
                av_free(picture);
161.
                return NULL;
162.
163.
           avpicture_fill((AVPicture *)picture, picture_buf, (PixelFormat)pix_fmt, width, height
164.
           return picture;
165.
       }
166.
167.
       void openVideo()
168.
169.
           AVCodec *video codec:
170.
           /* find the video encoder */
171.
172.
           video_codec = avcodec_find_encoder(video_cc->codec_id);
           if (!video_codec) {
173.
174.
                fprintf(stderr, "codec not found\n");
175.
                exit(1);
176.
           }
177.
178.
            /* open the codec */
179.
           if (avcodec_open(video_cc, video_codec) < 0) {</pre>
180.
               fprintf(stderr, "could not open video codec\n");
181.
                exit(1);
182.
           }
183.
184.
           /* allocate the encoded raw picture */
185.
           picture = allocPicture(video_cc->pix_fmt, video_cc->width, video_cc->height);
186.
           if (!picture) {
187.
                fprintf(stderr, "Could not allocate picture\n");
188.
                exit(1);
189.
           }
       }
190.
191.
       /* prepare a dummy image */
192.
193.
       void fill_yuv_image(AVFrame *pict, int frame_index, int width, int height)
194.
195.
           int x, y, i;
196.
197.
           i = frame_index;
198.
199.
            /* Y */
           for(y=0;y<height;y++) {</pre>
200.
201.
               for(x=0;x<width;x++) {</pre>
202.
                    pict->data[0][y * pict->linesize[0] + x] = x + y + i * 3;
203.
                }
204.
           }
205.
206.
           /* Cb and Cr */
207.
           for(y=0;y<height/2;y++) {
208.
                for(x=0;x<width/2;x++) {
209.
                   pict->data[1][y * pict->linesize[1] + x] = 128 + y + i * 2;
                    pict->data[2][y * pict->linesize[2] + x] = 64 + x + i * 5;
210.
```

```
211.
 212.
            }
 213.
        }
 214.
 215.
        void ffmpegInit()
 216.
 217.
            // initialize libavcodec, and register all codecs and formats
 218.
            av_register_all();
 219.
 220.
            // create a codec context
 221.
            createCodecContext();
 222.
 223.
            // open H.264 codec
 224.
            openVideo();
 225.
        }
 226.
 227.
        void getEncodedFrame(unsigned char *buffer, int& len)
 228.
 229.
            int out size:
 230.
 231.
            fill_yuv_image(picture, frame_count, video_cc->width, video_cc->height);
 232.
 233.
            // encode the frame
 234.
            out_size = avcodec_encode_video(video_cc, buffer, wrap_size-
        sizeof(FrameHeader), picture);
 235.
 236.
            len = out size;
 237.
            frame_count++;
        }
 238.
 239.
 240.
        int main()
 241.
        {
 242.
            unsigned char *send_outbuf;
 243.
            unsigned char *video_part;
 244.
 245.
            frame count = 0;
 246.
            wrap\_size = 20000;
 247.
            send_outbuf = (unsigned char *)malloc(wrap_size);
 248.
 249.
            // copy cmdHeader to frameInfo
 250.
            memcpy(frameHeader.cmdHeader,CMD_HEADER_STR,CMD_HEADER_LEN);
 251.
 252.
            memcpy(send_outbuf, &frameHeader, sizeof(FrameHeader));
 253.
            video_part = send_outbuf + sizeof(FrameHeader);
 254.
            ffmpeqInit();
 255.
 256.
            rtpInit();
 257.
 258.
            while (1)
 259.
 260.
                 int frame_len;
                // get encode frame
 261.
 262.
                getEncodedFrame(video_part, frame_len);
 263.
 264.
                printf("encodecFrame length is : %d\n", frame_len);
 265.
 266.
                 if (frame_len > 0)
 267.
                 {
 268.
                     rtpSend(send_outbuf, frame_len);
 269.
            }
 270.
 271.
 272.
            rtp_session_destroy(rtp_session_mgr.rtp_session);
 273.
 274.
            free(send outbuf);
 275.
            // Give us some time
 276.
 277.
            Sleep(250);
 278.
 279.
            ortp_exit();
 280.
        }
1
```

更多

上一篇 Introduction to Sound Programming with ALSA(使用alsa API接口编程)

下一篇 编程之美--2.20(程序理解和时间分析)

主题推荐 操作系统 源代码 windows xp 视频

猜你在找

h.264 SODB RBSP EBSP的区别

分享一段H264视频和AAC音频的RTP封包代码

Unity Application Block 学习笔记 之 在VS2012中 配置

gdb 0x000000000 in ?? () 错误处理 windows下使用vs2010编译live555 windows下使用FFmpeg进行音频转换程序

busybox简介

Linux环境下写一个 简单的 makefile

FFMPEG源码分析: avformat_open_input() (媒体打开

编译libav(ffmpeg)库



查看评论

5楼 mengfanxinqq 2013-08-29 09:27发表



你好,这个QOSAddSocketToFlow failed to add a flow with error 87 错误您是怎么解决的?我现在也遇到了这个错误。。

4楼 xi52gian 2013-08-07 11:09发表



34是H263,怎么可能会发送成功?网上还多都写34.真不知道发贴的人是否测试过?oRtp里定义了H264的PayLoad但是没 有添加到ProFile里,要自己在外部添加,而34在ORtp库中已经定义是34了!

3楼 简单的逻辑 2013-06-07 11:42发表



在 VS2008 下编译报错呢

[cpp]

01. 错误 1 fatal error C1083: 无法打开包括文

> 件:"inttypes.h": No such file or directory d:\ortp_test\ffmpeg-0.8-win32dev\include\libavutil\common.h 31 video_recver

在网上下载了 inttypes.h 加入后,又有其他的问题,怎么解决呢?

Re: aiaxhe 2013-06-08 10:48发表



回复baiynui1983:谢谢反馈,打包时在ortp_test\ffmpeg-0.8-win32-dev\include目录中忘记添加inttypes.h和 stdint.h头文件。已经修改,并重新上传了代码。

Re: 简单的逻辑 2013-06-14 14:05发表



回复ajaxhe:是这个问题,我在网上下载这两个文件,也解决了问题。楼主真认真!

Re: 简单的逻辑 2013-06-14 14:06发表



回复baiynui1983: 我最近在搞移动视频监控这方面的项目,并且以后也想网这方面发 展,希望能向你学习,我的 QQ: 99951468

Re: ajaxhe 2013-06-14 14:57发表



回复baiynui1983:多交流,共同进步

2楼 kangear 2013-04-03 11:34发表



牛,学习学习。

1楼 江南烟雨 2012-09-07 08:53发表



顶一顶~.嘿嘿~~

发表评论

用户名: wzw0114 评论内容: 4>

提交

*以上用户言论只代表其个人观点,不代表CSDN网站的观点或立场

核心技术类目

全部主题 Java VPN Android iOS ERP IE10 Eclipse CRM JavaScript Ubuntu NFC WAP jQuery 数据库 BI HTML5 Spring Apache Hadoop .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP Spark HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320

京 ICP 证 070598 号

北京创新乐知信息技术有限公司 版权所有

江苏乐知网络技术有限公司 提供商务支持

Copyright © 1999-2014, CSDN.NET, All Rights Reserved