FFMPEG的使用

Scott.Zheng

2019/3/10

AVTrace
Surveillance Solutions

目　录

一、编译

- FFMPEG为开源项目，可以自由下载编译和使用。
- 后面关于FFMPEG的使用例程均以CentOS系统为平台，不同平台的编译有一定的差异，但使用接口保持一致。

- 下载： git clone git://source.ffmpeg.org/ffmpeg.git ffmpeg
- 下载完成后会在当前目录下生成一个ffmpeg子目录，里面即是ffmpeg项目的全部源码。

- 编译：进入ffmpeg子目录，运行./configure完成环境配置，再运行make，系统开始编译。如果需要安装到当前系统，编译完成后运行make install，相关的库将会被拷贝到/usr/lib目录下，头文件也会被拷贝到/usr/include目录下。

./configure配置

> FFMPEG包含了大量的编解码、MUX、DEMUX、PROTOCOL等模块，我们在实际应用中并不需要这么多的模块，这时候就需要修改FFMPEG的默认配置了。输入./configure –help，可以显示有那些选项可以修改。

> 最常用的是—enable-shared，生成动态库；

> --disable-programs，不生成命令行程序，这在生成库时不需要；

> --disable-encoders、--disable-decoders，只做解码时可以把编码模块去掉，只做编码时可以把解码模块去掉。

> 根据自己的实际需要，删除不需要的模块，减小库的大小还可以使编译更容易通过。

二、编码

> 包含头文件（不同**FFMPEG**版本头文件有所差异）

> #include "avcodec/types.h"

> #include “avcodec/avcodec.h”

> 定义变量

> AVCodec *codec;

> AVCodecContext *c= NULL;

> AVFrame *picture;

> BYTE pbuf[176*144*3/2],outbuf[176*144*3/2];

> int ret,ii;

找到编码模块，申请资源，配置编码参数

```
codec = avcodec_find_encoder( CODEC_ID_H263 );
if( codec == NULL )          return ;
c = avcodec_alloc_context();
picture = avcodec_alloc_frame();
c->bit_rate                  = 100000;
c->width                     = 176;
c->height                    = 144;
c->time_base.num             = 1;
c->time_base.den             = 25;
c->gop_size                  = 12;
//c->max_b_frames            = 1;
c->pix_fmt                   = PIX_FMT_YUV420P;
```

打开编码模块，开始编码

```
if( avcodec_open( c, codec ) < 0 )      return;


 picture->data[0] = pbuf;
 picture->data[1] = picture->data[0] + 176*144;
 picture->data[2] = picture->data[1] + 176*144 / 4;
 picture->linesize[0] = c->width;
 picture->linesize[1] = c->width / 2;
 picture->linesize[2] = c->width / 2;
 for( ii=0; ii<50; ii++ )
{
    ret = avcodec_encode_video(  c,  outbuf,  176*144*3/2,  picture );
}
```

> 释放编码资源

> avcodec_close(c);

> av_free(c);

> av_free(picture);

三、解码

> 包含头文件（不同**FFMPEG**版本头文件有所差异）

> #include "avcodec/types.h"

> #include “avcodec/avcodec.h”

> 定义变量

> AVCodec *codec;

> AVCodecContext *c= NULL;

> AVFrame *picture;

> AVPacket avpkt;

> BYTE pbuf[176*144*3/2],outbuf[176*144*3/2];

> int ret,gotpiture;

找到解码模块、分配资源、打开解码模块

```
codec = avcodec_find_decoder( CODEC_ID_H264 );
av_init_packet( &avpkt );
c = avcodec_alloc_context();
picture = avcodec_alloc_frame();
ret = avcodec_open( c, codec );
avpkt.data = pbuf;
avpkt.size = 10000;
ret = avcodec_decode_video2( c, picture, &got_picture, &avpkt );
if( got_picture )
{
    picture->data[]          // 解码输出缓冲区
    picture->linesize[]      // 输出缓冲区大小
    c->width/height;         // 图像宽度/高度
}
```

释放解码资源

- avcodec_close(c);
- av_free(c);
- av_free(picture);
- av_free_packet(&avpkt);

四、去隔行

- AVPicture dstbuf;
- BYTE outbuf[176*144*3/2],

- dstbuf.data[0] = outbuff;
- dstbuf.data[1] = outbuff + 176*144;
- dstbuf.data[2] = outbuff + 176*144 + 176*144/4;

- dstbuf.linesize[0] = 176;
- dstbuf.linesize[1] = 88;
- dstbuf.linesize[2] = 88;

- avpicture_deinterlace(&dstbuf,(AVPicture*)picture,PIX_FMT_YUV420P,176,144);

五、缩放

```
struct SwsContext      *img_convert_ctx;
int                    sws_flags;
struct SwsContext      *sws_opts;
img_convert_ctx        = NULL;
sws_opts               = NULL;
AVPicture              srcbuf,dstbuf;
BYTE                   outbuf[176*144*3],
if( img_convert_ctx == NULL )
{
        if( sws_opts == NULL )
        {
            sws_flags = SWS_BICUBIC;
            sws_opts = sws_getContext( 176, 144, PIX_FMT_YUV420P, 176, 144, PIX_FMT_RGB32,
                sws_flags, NULL,NULL,NULL);
        }
        sws_flags = (int)av_get_int( sws_opts, "sws_flags", NULL );
        img_convert_ctx = sws_getCachedContext( img_convert_ctx, 176, 144, PIX_FMT_YUV420P, 176, 144,
            PIX_FMT_RGB32, sws_flags, NULL, NULL, NULL);
}
```

```
>   srcbuf.data[0] += (144-1)*srcbuf.linesize[0];
>   srcbuf.data[1] += (72-1)*srcbuf.linesize[1];
>   srcbuf.data[2] += (72-1)*srcbuf.linesize[2];
>   srcbuf.linesize[0] = -srcbuf.linesize[0];
>   srcbuf.linesize[1] = -srcbuf.linesize[1];
>   srcbuf.linesize[2] = -srcbuf.linesize[2];

>   dstbuf.data[0] = outbuff;
>   dstbuf.linesize[0] = 144*3;

>   sws_scale( img_convert_ctx, srcbuf.data, srcbuf.linesize, 0, 144, dstbuf.data, dstbuf.linesize );

>   if( img_convert_ctx )          sws_freeContext( img_convert_ctx );
>   if( sws_opts )                 av_free( sws_opts );
```

六、RTSP接口

Thank you for your attention

2019/3/10

AVTrace
Surveillance Solutions