

原 最简单的基于FFmpeg的AVUtil例子 (AVLog, AVOption等)

2015年07月18日 15:53:43 阅读数：11216

本文的示例程序记录了FFmpeg的libavutil中几种工具函数的使用方法：

AVLog：日志输出

AVOption (AVClass)：选项设置

AVDictionary：键值对存储

ParseUtil：字符串解析

几个libavutil的工具

AVLog

AVLog是FFmpeg的日志输出工具。在FFmpeg中所有的日志输出不是通过printf()函数而是通过av_log()函数。av_log()会最终调用fprintf(stderr,...)函数将日志内容输出到命令行界面上。但是在一些非命令行程序（MFC程序，Android程序等）中，av_log()调用的fprintf(stderr,...)就无法将日志内容显示出来了。对于这种情况，FFmpeg提供了日志回调函数av_log_set_callback()。该函数可以指定一个自定义的日志输出函数，将日志输出到指定的位置。

下面的自定义函数custom_output()将日志输出到了“simplest_ffmpeg_log.txt”文本中。

```
[cpp] 复制
1. void custom_output(void* ptr, int level, const char* fmt, va_list vl){
2.     FILE *fp = fopen("simplest_ffmpeg_log.txt", "a+");
3.     if(fp){
4.         fprintf(fp, fmt, vl);
5.         fflush(fp);
6.         fclose(fp);
7.     }
8. }
```

在主函数中调用av_log_set_callback()设置一下该函数就可以了，如下所示。

```
[cpp] 复制
1. int main(int argc, char* argv[])
2. {
3.     av_log_set_callback(custom_output);
4.     return 0;
5. }
```

此外，日志信息从重到轻分为Panic、Fatal、Error、Warning、Info、Verbose、Debug几个级别。下面的函数输出了几种不同级别的日志。

```
[cpp] 复制
1. void test_log(){
2.     av_register_all();
3.     AVFormatContext *obj=NULL;
4.     obj=avformat_alloc_context();
5.     printf("=====\n");
6.     av_log(obj, AV_LOG_PANIC, "Panic: Something went really wrong and we will crash now.\n");
7.     av_log(obj, AV_LOG_FATAL, "Fatal: Something went wrong and recovery is not possible.\n");
8.     av_log(obj, AV_LOG_ERROR, "Error: Something went wrong and cannot losslessly be recovered.\n");
9.     av_log(obj, AV_LOG_WARNING, "Warning: This may or may not lead to problems.\n");
10.    av_log(obj, AV_LOG_INFO, "Info: Standard information.\n");
11.    av_log(obj, AV_LOG_VERBOSE, "Verbose: Detailed information.\n");
12.    av_log(obj, AV_LOG_DEBUG, "Debug: Stuff which is only useful for libav* developers.\n");
13.    printf("=====\n");
14.    avformat_free_context(obj);
15. }
```

PS：该部分源代码的解析可以参考文章《[FFmpeg源代码简单分析：日志输出系统（av_log\(\)等）](#)》

AVOption (AVClass)

AVOption是FFmpeg的选项设置工具。与AVOption最相关的选项设置函数就是av_opt_set()了。AVOption的核心概念就是“根据字符串操作结构体的属性值”。例如下面代码中“#if”和“#else”之间代码的作用和“#else”和“#endif”之间代码的作用是一样的。

```

1.  #if TEST_OPT
2.      av_opt_set(pCodecCtx,"b","400000",0);          //bitrate
3.      //Another method
4.      //av_opt_set_int(pCodecCtx,"b",400000,0);      //bitrate
5.
6.      av_opt_set(pCodecCtx,"time_base","1/25",0); //time_base
7.      av_opt_set(pCodecCtx,"bf","5",0);           //max b frame
8.      av_opt_set(pCodecCtx,"g","25",0);           //gop
9.      av_opt_set(pCodecCtx,"qmin","10",0);         //qmin/qmax
10.     av_opt_set(pCodecCtx,"qmax","51",0);
11. #else
12.     pCodecCtx->time_base.num = 1;
13.     pCodecCtx->time_base.den = 25;
14.     pCodecCtx->max_b_frames=5;
15.     pCodecCtx->bit_rate = 400000;
16.     pCodecCtx->gop_size=25;
17.     pCodecCtx->qmin = 10;
18.     pCodecCtx->qmax = 51;
19. #endif

```

同理，av_opt_get()可以将结构体的属性值以字符串的形式返回回来。例如下面这段代码就验证了av_opt_get()的作用：

```

1.  char *val_str=(char *)av_malloc(50);
2.
3.  //preset: ultrafast, superfast, veryfast, faster, fast,
4.  //medium, slow, slower, veryslow, placebo
5.  av_opt_set(pCodecCtx->priv_data,"preset","slow",0);
6.  //tune: film, animation, grain, stillimage, psnr,
7.  //ssim, fastdecode, zerolatency
8.  av_opt_set(pCodecCtx->priv_data,"tune","zerolatency",0);
9.  //profile: baseline, main, high, high10, high422, high444
10. av_opt_set(pCodecCtx->priv_data,"profile","main",0);
11.
12. //print
13. av_opt_get(pCodecCtx->priv_data,"preset",0,(uint8_t **)&val_str);
14. printf("preset val: %s\n",val_str);
15. av_opt_get(pCodecCtx->priv_data,"tune",0,(uint8_t **)&val_str);
16. printf("tune val: %s\n",val_str);
17. av_opt_get(pCodecCtx->priv_data,"profile",0,(uint8_t **)&val_str);
18. printf("profile val: %s\n",val_str);
19. av_free(val_str);

```

可以通过av_opt_find()获取结构体中任意选项的AVOption结构体。写了一个简单的函数读取该结构体中一些字段的值。

```

1. void print_opt(const AVOption *opt_test){
2.
3.     printf("=====\n");
4.     printf("Option Information:\n");
5.     printf("[name]s\n", opt_test->name);
6.     printf("[help]s\n", opt_test->help);
7.     printf("[offset]d\n", opt_test->offset);
8.
9.     switch(opt_test->type){
10.    case AV_OPT_TYPE_INT:{
11.        printf("[type]int\n[default]d\n", opt_test->default_val.i64);
12.        break;
13.    }
14.    case AV_OPT_TYPE_INT64:{
15.        printf("[type]int64\n[default]lld\n", opt_test->default_val.i64);
16.        break;
17.    }
18.    case AV_OPT_TYPE_FLOAT:{
19.        printf("[type]float\n[default]f\n", opt_test->default_val.dbl);
20.        break;
21.    }
22.    case AV_OPT_TYPE_STRING:{
23.        printf("[type]string\n[default]s\n", opt_test->default_val.str);
24.        break;
25.    }
26.    case AV_OPT_TYPE_RATIONAL:{
27.        printf("[type]rational\n[default]d/d\n", opt_test->default_val.q.num, opt_test->default_val.q.den);
28.        break;
29.    }
30.    default:{
31.        printf("[type]others\n");
32.        break;
33.    }
34. }
35.
36. printf("[max val]f\n", opt_test->max);
37. printf("[min val]f\n", opt_test->min);
38.
39. if(opt_test->flags&AV_OPT_FLAG_ENCODING_PARAM){
40.     printf("Encoding param.\n");
41. }
42. if(opt_test->flags&AV_OPT_FLAG_DECODING_PARAM){
43.     printf("Decoding param.\n");
44. }
45. if(opt_test->flags&AV_OPT_FLAG_AUDIO_PARAM){
46.     printf("Audio param.\n");
47. }
48. if(opt_test->flags&AV_OPT_FLAG_VIDEO_PARAM){
49.     printf("Video param.\n");
50. }
51. if(opt_test->unit!=NULL)
52.     printf("Unit belong to:s\n", opt_test->unit);
53.
54. printf("=====\n");
55. }

```

使用下列代码调用上面的函数就可以打印出AVOption结构体每个字段的值。

```

1. const AVOption *opt=NULL;
2. opt=av_opt_find(pCodecCtx, "b", NULL, 0, 0);
3. print_opt(opt);
4. opt=av_opt_find(pCodecCtx, "g", NULL, 0, 0);
5. print_opt(opt);
6. opt=av_opt_find(pCodecCtx, "time_base", NULL, 0, 0);
7. print_opt(opt);

```

下面代码可以打印出支持AVOption（即包含AVClass）的结构体的所有选项：

```

1. void list_obj_opt(void *obj){
2.     printf("Output some option info about object:\n");
3.     printf("Object name:%s\n", (*(AVClass **) obj)->class_name);
4.     printf("=====\n");
5.     printf("Video param:\n");
6.     av_opt_show2(obj, stderr, AV_OPT_FLAG_VIDEO_PARAM, NULL);
7.     printf("Audio param:\n");
8.     av_opt_show2(obj, stderr, AV_OPT_FLAG_AUDIO_PARAM, NULL);
9.     printf("Decoding param:\n");
10.    av_opt_show2(obj, stderr, AV_OPT_FLAG_DECODING_PARAM, NULL);
11.    printf("Encoding param:\n");
12.    av_opt_show2(obj, stderr, AV_OPT_FLAG_ENCODING_PARAM, NULL);
13.    printf("=====\n");
14. }

```

下面代码调用上面的函数就可以打印出AVFormatContext中的所有选项。

```
[cpp]
1. void test_opt(){
2.     av_register_all();
3.     AVFormatContext *obj=NULL;
4.     obj=avformat_alloc_context();
5.     list_obj_opt(obj);
6.     avformat_free_context(obj);
7. }
```

PS：该部分源代码的解析可以参考文章《[FFmpeg源代码简单分析：结构体成员管理系统-AVClass](#)》、《[FFmpeg源代码简单分析：结构体成员管理系统-AVOption](#)》。

AVDictionary

AVDictionary是FFmpeg的键值对存储工具，FFmpeg经常使用AVDictionary设置/读取内部参数。下面这段代码记录了AVDictionary的使用方法。



```
[cpp]
1. void test_avdictionary(){
2.
3.     AVDictionary *d = NULL;
4.     AVDictionaryEntry *t = NULL;
5.
6.     av_dict_set(&d, "name", "lei xiaohua", 0);
7.     av_dict_set(&d, "email", "leixiaohua1020@126.com", 0);
8.     av_dict_set(&d, "school", "cuc", 0);
9.     av_dict_set(&d, "gender", "man", 0);
10.    av_dict_set(&d, "website", "http://blog.csdn.net/leixiaohua1020", 0);
11.    //av_strdup()
12.    char *k = av_strdup("location");
13.    char *v = av_strdup("Beijing-China");
14.    av_dict_set(&d, k, v, AV_DICT_DONT_STRDUP_KEY | AV_DICT_DONT_STRDUP_VAL);
15.    printf("=====\n");
16.    int dict_cnt= av_dict_count(d);
17.    printf("dict_count:%d\n",dict_cnt);
18.    printf("dict_element:\n");
19.    while (t = av_dict_get(d, "", t, AV_DICT_IGNORE_SUFFIX)) {
20.        printf("key:%10s | value:%s\n",t->key,t->value);
21.    }
22.
23.    t = av_dict_get(d, "email", t, AV_DICT_IGNORE_SUFFIX);
24.    printf("email is %s\n",t->value);
25.    printf("=====\n");
26.    av_dict_free(&d);
27. }
```

ParseUtil

ParseUtil是FFmpeg的字符串解析工具。它的分辨率解析函数av_parse_video_size()可以从形如"1920x1080"的字符串中解析出图像宽为1920，高为1080；它的帧率函数av_parse_video_rate()可以解析出帧率信息；它的时间解析函数则可以形如"00:01:01"的字符串解析出时间的毫秒数。下面这段代码记录了ParseUtil的使用方法。

```
[cpp]
1. void test_parseutil(){
2.     char input_str[100]={0};
3.     printf("=====\n");
4.     int output_w=0;
5.     int output_h=0;
6.     strcpy(input_str,"1920x1080");
7.     av_parse_video_size(&output_w,&output_h,input_str);
8.     printf("w:%4d | h:%4d\n",output_w,output_h);
9.     strcpy(input_str,"vga");
10.    //strcpy(input_str,"hd1080");
11.    //strcpy(input_str,"ntsc");
12.    av_parse_video_size(&output_w,&output_h,input_str);
13.    printf("w:%4d | h:%4d\n",output_w,output_h);
14.    printf("=====\n");
15.    AVRational output_rational={0,0};
16.    strcpy(input_str,"15/1");
17.    av_parse_video_rate(&output_rational,input_str);
18.    printf("framerate:%d/%d\n",output_rational.num,output_rational.den);
19.    strcpy(input_str,"pal");
20.    av_parse_video_rate(&output_rational,input_str);
21.    printf("framerate:%d/%d\n",output_rational.num,output_rational.den);
22.    printf("=====\n");
23.    int64_t output_timeval;
24.    strcpy(input_str,"00:01:01");
25.    av_parse_time(&output_timeval,input_str,1);
26.    printf("microseconds:%lld\n",output_timeval);
27.    printf("=====\n");
28. }
```

源代码

```
[cpp]    
1.  /**  
2.   * 最简单的FFmpeg的AVUtil示例  
3.   * Simplest FFmpeg AVUtil  
4.   *  
5.   * 雷霄骅 Lei Xiaohua  
6.   * leixiaohua1020@126.com  
7.   * 中国传媒大学/数字电视技术  
8.   * Communication University of China / Digital TV Technology  
9.   * http://blog.csdn.net/leixiaohua1020  
10.  *  
11.  * 本程序是FFmpeg中的libavutil的示例，目前包含：  
12.  * AVLog  
13.  * AVOption (AVClass)  
14.  * AVDictionary  
15.  * ParseUtil  
16.  *  
17.  * This software is the example about FFmpeg's libavutil.  
18.  * It contains:  
19.  * AVLog  
20.  * AVOption (AVClass)  
21.  * AVDictionary  
22.  * ParseUtil  
23.  *  
24.  */  
25.  
26. #include <stdio.h>  
27.  
28. #define __STDC_CONSTANT_MACROS  
29.  
30. #ifdef WIN32  
31. //Windows  
32. extern "C"  
33. {  
34. #include "libavcodec/avcodec.h"  
35. #include "libavformat/avformat.h"  
36. #include "libavutil/opt.h"  
37. #include "libavutil/parseutils.h"  
38. #include "libavutil/avutil.h"  
39. };  
40. #else  
41. //Linux...  
42. #ifdef __cplusplus  
43. extern "C"  
44. {  
45. #endif  
46. #include <libavcodec/avcodec.h>  
47. #include <libavformat/avformat.h>  
48. #include <libavutil/opt.h>  
49. #include <libavutil/parseutils.h>  
50. #include <libavutil/avutil.h>  
51. #ifdef __cplusplus  
52. };  
53. #endif  
54. #endif  
55.  
56.  
57. #define TEST_OPT 1  
58. #define TEST_LOG 1  
59. #define TEST_DIC 0  
60.  
61.  
62. void list_obj_opt(void *obj){  
63.     printf("Output some option info about object:\n");  
64.     printf("Object name:%s\n", (*(AVClass **) obj)->class_name);  
65.     printf("=====\n");  
66.     printf("Video param:\n");  
67.     av_opt_show2(obj, stderr, AV_OPT_FLAG_VIDEO_PARAM, NULL);  
68.     printf("Audio param:\n");  
69.     av_opt_show2(obj, stderr, AV_OPT_FLAG_AUDIO_PARAM, NULL);  
70.     printf("Decoding param:\n");  
71.     av_opt_show2(obj, stderr, AV_OPT_FLAG_DECODING_PARAM, NULL);  
72.     printf("Encoding param:\n");  
73.     av_opt_show2(obj, stderr, AV_OPT_FLAG_ENCODING_PARAM, NULL);  
74.     printf("=====\n");  
75. }  
76.  
77. void test_opt(){  
78.     av_register_all();  
79.     AVFormatContext *obj=NULL;  
80.     obj=avformat_alloc_context();  
81.     list_obj_opt(obj);  
82.     avformat_free_context(obj);
```

```

83. }
84.
85.
86. void test_log(){
87.     av_register_all();
88.     AVFormatContext *obj=NULL;
89.     obj=avformat_alloc_context();
90.     printf("=====\n");
91.     av_log(obj,AV_LOG_PANIC,"Panic: Something went really wrong and we will crash now.\n");
92.     av_log(obj,AV_LOG_FATAL,"Fatal: Something went wrong and recovery is not possible.\n");
93.     av_log(obj,AV_LOG_ERROR,"Error: Something went wrong and cannot losslessly be recovered.\n");
94.     av_log(obj,AV_LOG_WARNING,"Warning: This may or may not lead to problems.\n");
95.     av_log(obj,AV_LOG_INFO,"Info: Standard information.\n");
96.     av_log(obj,AV_LOG_VERBOSE,"Verbose: Detailed information.\n");
97.     av_log(obj,AV_LOG_DEBUG,"Debug: Stuff which is only useful for libav* developers.\n");
98.     printf("=====\n");
99.     avformat_free_context(obj);
100. }
101.
102. void print_opt(const AVOption *opt_test){
103.
104.     printf("=====\n");
105.     printf("Option Information:\n");
106.     printf("[name]s\n",opt_test->name);
107.     printf("[help]s\n",opt_test->help);
108.     printf("[offset]d\n",opt_test->offset);
109.
110.     switch(opt_test->type){
111.     case AV_OPT_TYPE_INT:{
112.         printf("[type]int\n[default]d\n",opt_test->default_val.i64);
113.         break;
114.     }
115.     case AV_OPT_TYPE_INT64:{
116.         printf("[type]int64\n[default]lld\n",opt_test->default_val.i64);
117.         break;
118.     }
119.     case AV_OPT_TYPE_FLOAT:{
120.         printf("[type]float\n[default]f\n",opt_test->default_val.dbl);
121.         break;
122.     }
123.     case AV_OPT_TYPE_STRING:{
124.         printf("[type]string\n[default]s\n",opt_test->default_val.str);
125.         break;
126.     }
127.     case AV_OPT_TYPE_RATIONAL:{
128.         printf("[type]rational\n[default]d/d\n",opt_test->default_val.q.num,opt_test->default_val.q.den);
129.         break;
130.     }
131.     default:{
132.         printf("[type]others\n");
133.         break;
134.     }
135. }
136.
137.     printf("[max val]f\n",opt_test->max);
138.     printf("[min val]f\n",opt_test->min);
139.
140.     if(opt_test->flags&AV_OPT_FLAG_ENCODING_PARAM){
141.         printf("Encoding param.\n");
142.     }
143.     if(opt_test->flags&AV_OPT_FLAG_DECODING_PARAM){
144.         printf("Decoding param.\n");
145.     }
146.     if(opt_test->flags&AV_OPT_FLAG_AUDIO_PARAM){
147.         printf("Audio param.\n");
148.     }
149.     if(opt_test->flags&AV_OPT_FLAG_VIDEO_PARAM){
150.         printf("Video param.\n");
151.     }
152.     if(opt_test->unit!=NULL)
153.         printf("Unit belong to:%s\n",opt_test->unit);
154.
155.     printf("=====\n");
156. }
157.
158. int flush_encoder(AVFormatContext *fmt_ctx,unsigned int stream_index)
159. {
160.     int ret;
161.     int got_frame;
162.     AVPacket enc_pkt;
163.     if (!(fmt_ctx->streams[stream_index]->codec->capabilities &
164.         CODEC_CAP_DELAY))
165.         return 0;
166.     while (1) {
167.         printf("Flushing stream #%u encoder\n", stream_index);
168.         enc_pkt.data = NULL;
169.         enc_pkt.size = 0;
170.         av_init_packet(&enc_pkt);
171.         ret = avcodec_encode_video2 (fmt_ctx->streams[stream_index]->codec, &enc_pkt,
172.             NULL, &got_frame);
173.         av_frame_free(NULL);

```

```

174.         if (ret < 0)
175.             break;
176.         if (!got_frame){
177.             ret=0;
178.             break;
179.         }
180.         printf("Succeed to encode 1 frame!\n");
181.         /* mux encoded frame */
182.         ret = av_write_frame(fmt_ctx, &enc_pkt);
183.         if (ret < 0)
184.             break;
185.     }
186.     return ret;
187. }
188.
189. int encoder(){
190.     AVFormatContext* pFormatCtx;
191.     AVStream* video_st;
192.     AVCodecContext* pCodecCtx;
193.     AVCodec* pCodec;
194.
195.     uint8_t* picture_buf;
196.     AVFrame* picture;
197.     int size;
198.     int ret;
199.     AVPacket pkt;
200.     int y_size;
201.
202.     FILE *in_file = fopen("ds_480x272.yuv", "rb"); //Input YUV data
203.     int in_w=480,in_h=272; //Input width and height
204.     //Frames to encode
205.     int framenum=100;
206.     const char* out_file = "ds.h264"; //Output Filepath
207.     //const char* out_file = "ds.ts";
208.     //const char* out_file = "ds.hevc";
209.     char temp_str[250]={0};
210.
211.     av_register_all();
212.
213.     avformat_alloc_output_context2(&pFormatCtx, NULL, NULL, out_file);
214.
215.     if (avio_open(&pFormatCtx->pb,out_file, AVIO_FLAG_READ_WRITE) < 0){
216.         printf("Failed to open output file!\n");
217.         return -1;
218.     }
219.
220.     pCodec = avcodec_find_encoder(pFormatCtx->oformat->video_codec);
221.     if (!pCodec) {
222.         fprintf(stderr, "Codec not found.\n");
223.         return -1;
224.     }
225.     video_st = avformat_new_stream(pFormatCtx, pCodec);
226.     video_st->time_base.num = 1;
227.     video_st->time_base.den = 25;
228.
229.     if (video_st==NULL){
230.         return -1;
231.     }
232.     //Param that must set
233.     pCodecCtx = video_st->codec;
234.     pCodecCtx->codec_type = AVMEDIA_TYPE_VIDEO;
235.     pCodecCtx->pix_fmt = PIX_FMT_YUV420P;
236.     pCodecCtx->width = in_w;
237.     pCodecCtx->height = in_h;
238.
239.     #if TEST_OPT
240.     av_opt_set(pCodecCtx,"b","400000",0); //bitrate
241.     //Another method
242.     //av_opt_set_int(pCodecCtx,"b",400000,0); //bitrate
243.
244.     av_opt_set(pCodecCtx,"time_base","1/25",0); //time_base
245.     av_opt_set(pCodecCtx,"bf","5",0); //max b frame
246.     av_opt_set(pCodecCtx,"g","25",0); //gop
247.     av_opt_set(pCodecCtx,"qmin","10",0); //qmin/qmax
248.     av_opt_set(pCodecCtx,"qmax","51",0);
249.     #else
250.     pCodecCtx->time_base.num = 1;
251.     pCodecCtx->time_base.den = 25;
252.     pCodecCtx->max_b_frames=5;
253.     pCodecCtx->bit_rate = 400000;
254.     pCodecCtx->gop_size=25;
255.     pCodecCtx->qmin = 10;
256.     pCodecCtx->qmax = 51;
257.     #endif
258.
259.     #if TEST_OPT
260.     //list_obj_opt(pFormatCtx);
261.     //list_obj_opt(pCodecCtx);
262.     const AVOption *opt=NULL;
263.     opt=av_opt_find(pCodecCtx, "b", NULL, 0, 0);
264.     print_opt(opt);

```

```

265.     opt=av_opt_find(pCodecCtx, "g", NULL, 0, 0);
266.     print_opt(opt);
267.     opt=av_opt_find(pCodecCtx, "time_base", NULL, 0, 0);
268.     print_opt(opt);
269.     //Get Option
270.     //Get String
271.     int64_t *val_str=(int64_t *)av_malloc(1*sizeof(int64_t));
272.     av_opt_get(pCodecCtx,"b",0,(uint8_t **)&val_str);
273.     printf("get bitrate(str):%s\n",val_str);
274.     av_free(val_str);
275.     //Get int
276.     int64_t val_int=0;
277.     av_opt_get_int(pCodecCtx,"b",0,&val_int);
278.     printf("get bitrate(int):%lld\n",val_int);
279. #endif
280.
281.     AVDictionary *param = 0;
282.
283.     //H.264
284.     if(pCodecCtx->codec_id == AV_CODEC_ID_H264) {
285.         char *val_str=(char *)av_malloc(50);
286.         //List it
287.         //list_obj_opt(pCodecCtx->priv_data);
288.
289.         //preset: ultrafast, superfast, veryfast, faster, fast,
290.         //medium, slow, slower, veryslow, placebo
291.         av_opt_set(pCodecCtx->priv_data,"preset","slow",0);
292.         //tune: film, animation, grain, stillimage, psnr,
293.         //ssim, fastdecode, zerolatency
294.         av_opt_set(pCodecCtx->priv_data,"tune","zerolatency",0);
295.         //profile: baseline, main, high, high10, high422, high444
296.         av_opt_set(pCodecCtx->priv_data,"profile","main",0);
297.
298.         //print
299.         av_opt_get(pCodecCtx->priv_data,"preset",0,(uint8_t **)&val_str);
300.         printf("preset val: %s\n",val_str);
301.         av_opt_get(pCodecCtx->priv_data,"tune",0,(uint8_t **)&val_str);
302.         printf("tune val: %s\n",val_str);
303.         av_opt_get(pCodecCtx->priv_data,"profile",0,(uint8_t **)&val_str);
304.         printf("profile val: %s\n",val_str);
305.         av_free(val_str);
306.
307.     #if TEST_DIC
308.         av_dict_set(&m, "preset", "slow", 0);
309.         av_dict_set(&m, "tune", "zerolatency", 0);
310.         //av_dict_set(&m, "profile", "main", 0);
311.     #endif
312.     }
313.     //H.265
314.     if(pCodecCtx->codec_id == AV_CODEC_ID_H265){
315.         //list_obj_opt(pCodecCtx->priv_data);
316.
317.         //preset: ultrafast, superfast, veryfast, faster, fast,
318.         //medium, slow, slower, veryslow, placebo
319.         av_opt_set(pCodecCtx->priv_data, "preset", "ultrafast", 0);
320.         //tune: psnr, ssim, zerolatency, fastdecode
321.         av_opt_set(pCodecCtx->priv_data, "tune", "zero-latency", 0);
322.         //profile: main, main10, mainstillpicture
323.         av_opt_set(pCodecCtx->priv_data,"profile","main",0);
324.     }
325.
326.     if (avcodec_open2(pCodecCtx, pCodec,&m) < 0){
327.         printf("Failed to open encoder!\n");
328.         return -1;
329.     }
330.
331.     picture = avcodec_alloc_frame();
332.     size = avpicture_get_size(pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height);
333.     picture_buf = (uint8_t *)av_malloc(size);
334.     avpicture_fill((AVPicture *)picture, picture_buf, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height);
335.
336.     //Write File Header
337.     avformat_write_header(pFormatCtx,NULL);
338.
339.     y_size = pCodecCtx->width * pCodecCtx->height;
340.     av_new_packet(&pkt,y_size*3);
341.
342.     for (int i=0; i<framenum; i++){
343.         //Read YUV
344.         if (fread(picture_buf, 1, y_size*3/2, in_file) < 0){
345.             printf("Failed to read YUV data!\n");
346.             return -1;
347.         }else if(feof(in_file)){
348.             break;
349.         }
350.         picture->data[0] = picture_buf;           // Y
351.         picture->data[1] = picture_buf+ y_size;   // U
352.         picture->data[2] = picture_buf+ y_size*5/4; // V
353.         //PTS
354.         picture->pts=i;
355.         int got_picture=0;

```



```

356.     //Encode
357.     ret = avcodec_encode_video2(pCodecCtx, &pkt,picture, &got_picture);
358.     if(ret < 0){
359.         printf("Failed to encode!\n");
360.         return -1;
361.     }
362.     if (got_picture==1){
363.         //printf("Succeed to encode 1 frame!\n");
364.         pkt.stream_index = video_st->index;
365.         ret = av_write_frame(pFormatCtx, &pkt);
366.         av_free_packet(&pkt);
367.     }
368. }
369. //Flush Encoder
370. ret = flush_encoder(pFormatCtx,0);
371. if (ret < 0) {
372.     printf("Flushing encoder failed\n");
373.     return -1;
374. }
375.
376. //Write file trailer
377. av_write_trailer(pFormatCtx);
378.
379. //Clean
380. if (video_st){
381.     avcodec_close(video_st->codec);
382.     av_free(picture);
383.     av_free(picture_buf);
384. }
385. avio_close(pFormatCtx->pb);
386. avformat_free_context(pFormatCtx);
387.
388. fclose(in_file);
389. return 0;
390. }
391.
392. void custom_output(void* ptr, int level, const char* fmt,va_list vl){
393.     FILE *fp = fopen("simplest_ffmpeg_log.txt","a+");
394.     if(fp){
395.         vfprintf(fp,fmt,vl);
396.         fflush(fp);
397.         fclose(fp);
398.     }
399. }
400.
401. void test_parseutil(){
402.     char input_str[100]={0};
403.     printf("===== Parse Video Size =====\n");
404.     int output_w=0;
405.     int output_h=0;
406.     strcpy(input_str,"1920x1080");
407.     av_parse_video_size(&output_w,&output_h,input_str);
408.     printf("w:%4d | h:%4d\n",output_w,output_h);
409.     strcpy(input_str,"vga");
410.     //strcpy(input_str,"hd1080");
411.     //strcpy(input_str,"ntsc");
412.     av_parse_video_size(&output_w,&output_h,input_str);
413.     printf("w:%4d | h:%4d\n",output_w,output_h);
414.     printf("===== Parse Frame Rate =====\n");
415.     AVRational output_rational={0,0};
416.     strcpy(input_str,"15/1");
417.     av_parse_video_rate(&output_rational,input_str);
418.     printf("framerate:%d/%d\n",output_rational.num,output_rational.den);
419.     strcpy(input_str,"pal");
420.     av_parse_video_rate(&output_rational,input_str);
421.     printf("framerate:%d/%d\n",output_rational.num,output_rational.den);
422.     printf("===== Parse Time =====\n");
423.     int64_t output_timeval;
424.     strcpy(input_str,"00:01:01");
425.     av_parse_time(&output_timeval,input_str,1);
426.     printf("microseconds:%lld\n",output_timeval);
427.     printf("===== \n");
428. }
429.
430. void test_avdictionary(){
431.
432.     AVDictionary *d = NULL;
433.     AVDictionaryEntry *t = NULL;
434.
435.     av_dict_set(&d, "name", "lei xiaohua", 0);
436.     av_dict_set(&d, "email", "leixiaohua1020@126.com", 0);
437.     av_dict_set(&d, "school", "cuc", 0);
438.     av_dict_set(&d, "gender", "man", 0);
439.     av_dict_set(&d, "website", "http://blog.csdn.net/leixiaohua1020", 0);
440.     //av_strdup()
441.     char *k = av_strdup("location");
442.     char *v = av_strdup("Beijing-China");
443.     av_dict_set(&d, k, v, AV_DICT_DONT_STRDUP_KEY | AV_DICT_DONT_STRDUP_VAL);
444.     printf("===== \n");
445.     int dict_cnt= av_dict_count(d);
446.     printf("dict_count:%d\n",dict_cnt);
447.     //printf("dict_count:%d\n",dict_cnt);

```

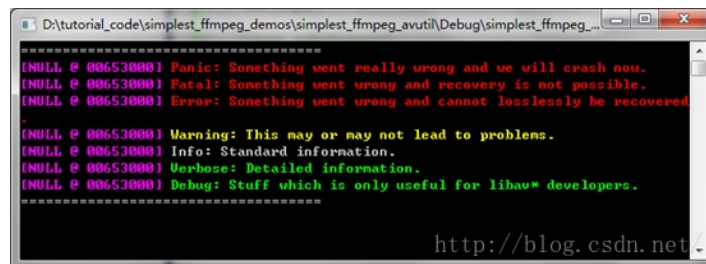
```

447.     printf("dict_element:\n");
448.     while (t = av_dict_get(d, "", t, AV_DICT_IGNORE_SUFFIX)) {
449.         printf("key:%10s | value:%s\n",t->key,t->value);
450.     }
451.
452.     t = av_dict_get(d, "email", t, AV_DICT_IGNORE_SUFFIX);
453.     printf("email is %s\n",t->value);
454.     printf("=====\n");
455.     av_dict_free(&d);
456. }
457.
458. int main(int argc, char* argv[])
459. {
460.     int loglevel=av_log_get_level();
461.     av_log_set_level(AV_LOG_DEBUG);
462.     //av_log_set_flags(AV_LOG_PRINT_LEVEL);
463.     //av_log_set_callback(custom_output);
464.     test_log();
465.
466.     test_avdictionary();
467.     test_parseutil();
468.
469.     //test_opt();
470.
471.     encoder();
472.
473.     return 0;
474. }

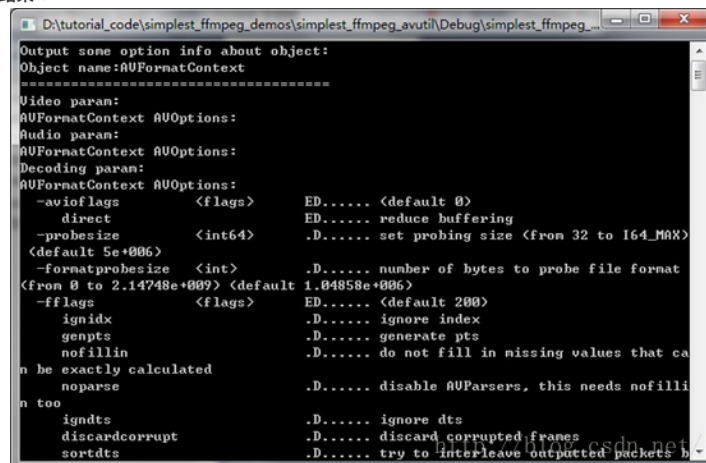
```

运行结果

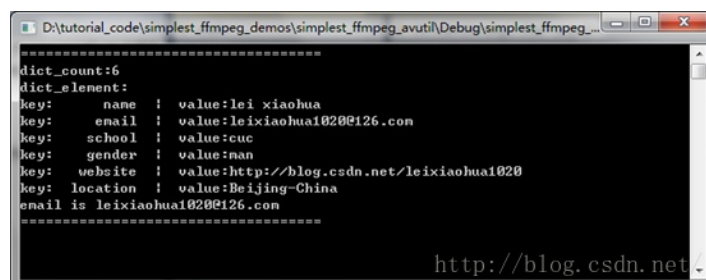
不同级别的AVLog日志输出后的结果：



结构体中所有AVOption信息输出后的结果：



AVDictionary示例输出的结果：



ParseUtil示例输出的结果：

```
D:\tutorial_code\simplest_ffmpeg_demo\simplest_ffmpeg_avutil\Debug\simplest_ffmpeg_...
===== Parse Video Size =====
w:1920 : h:1080
w: 640 : h: 480
===== Parse Frame Rate =====
framerate:15/1
framerate:25/1
===== Parse Time =====
microseconds:61000000
=====
```

编码的时候设置回调函数后输出到文本中的日志：

```
simplest_ffmpeg_log.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
frame= 91 QP=22.89 NAL=2 Slice:P Poc:6 I:3 P:380 SKIP:127 size=1911 bytes
frame= 92 QP=24.07 NAL=2 Slice:B Poc:4 I:0 P:198 SKIP:306 size=442 bytes
frame= 93 QP=25.22 NAL=0 Slice:B Poc:2 I:0 P:225 SKIP:283 size=375 bytes
frame= 94 QP=22.45 NAL=2 Slice:P Poc:12 I:3 P:439 SKIP:68 size=3160 bytes
frame= 95 QP=23.77 NAL=2 Slice:B Poc:10 I:0 P:230 SKIP:272 size=550 bytes
frame= 96 QP=26.11 NAL=0 Slice:B Poc:8 I:0 P:222 SKIP:279 size=462 bytes
frame= 97 QP=22.41 NAL=2 Slice:P Poc:18 I:10 P:401 SKIP:99 size=4399 bytes
frame= 98 QP=24.78 NAL=2 Slice:B Poc:16 I:0 P:236 SKIP:236 size=1580 bytes
frame= 99 QP=25.13 NAL=0 Slice:B Poc:14 I:0 P:231 SKIP:262 size=781 bytes
frame I:5 Avg QP:25.40 size: 11118
frame P:25 Avg QP:23.28 size: 3680
frame B:70 Avg QP:26.70 size: 668
consecutive B-frames: 6.0% 4.0% 24.0% 32.0% 10.0% 24.0%
mb I 116..4: 26.7% 0.0% 73.3%
mb P 116..4: 0.6% 0.0% 0.7% P16..4: 44.1% 24.9% 13.7% 0.0% 0.0% skip:16.0%
mb B 116..4: 0.0% 0.0% 0.0% B16..8: 34.8% 6.3% 1.4% direct: 1.4% skip:56.1%
final ratefactor: 21.01
direct mvvs spatial:97.1% temporal:2.9%
coded y,uvDC,uvAC intra: 71.0% 83.6% 48.4% inter: 9.4% 12.6% 0.7%
i16 v,h,dc,p: 12% 33% 6% 49%
i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 10% 8% 10% 9% 10% 11% 12% 12% 17%
i8c dc,h,v,p: 44% 22% 20% 13%
Weighted P-Frames: Y:0.0% UV:0.0%
ref P L0: 66.5% 15.1% 13.6% 2.7% 1.8% 0.4%
ref B L0: 92.2% 6.6% 0.9% 0.3%
ref B L1: 93.0% 7.0%
kb/s:388.64
Statistics: 0 seeks, 100 writeouts
```

下载

Simplest Ffmpeg AVUtil

项目主页

SourceForge：<https://sourceforge.net/projects/simplestffmpegavutil/>

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

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

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8924311>

本程序是Ffmpeg中的libavutil的示例，目前包含：

- AVLog
- AVOption (AVClass)
- AVDictionary
- ParseUtil

版权声明：本文为博主原创文章，未经博主允许不得转载。<https://blog.csdn.net/leixiaohua1020/article/details/46890739>

文章标签：[FFmpeg](#) [AVLog](#) [AVOption](#) [AVDictionary](#) [ParseUtil](#)

个人分类：[FFMPEG](#)

所属专栏：[FFmpeg](#)

此PDF由spyyg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com