

zmqsend.c

Go to the documentation of this file.

```

1  /*
2  * Copyright (c) 2013 Stefano Sabatini
3  *
4  * This file is part of FFmpeg.
5  *
6  * FFmpeg is free software; you can redistribute it and/or
7  * modify it under the terms of the GNU Lesser General Public
8  * License as published by the Free Software Foundation; either
9  * version 2.1 of the License, or (at your option) any later version.
10 *
11 * FFmpeg is distributed in the hope that it will be useful,
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
14 * Lesser General Public License for more details.
15 *
16 * You should have received a copy of the GNU Lesser General Public
17 * License along with FFmpeg; if not, write to the Free Software
18 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
19 */
20
21 #include "config.h"
22
23 #include <stdio.h>
24 #include <string.h>
25 #include <zmq.h>
26
27 #include "libavutil/log.h"
28 #include "libavutil/mem.h"
29 #include "libavutil/bprint.h"
30
31 #if HAVE_UNISTD_H
32 #include <unistd.h>          /* getopt */
33 #endif
34
35 #if !HAVE_GETOPT
36 #include "compat/getopt.c"
37 #endif
38
39 /**
40 * @file
41 * zmq message sender example, meant to be used with the zmq filters
42 */
43
44 static void usage(void)
45 {
46     printf("send message to ZMQ recipient, to use with the zmq filters\n");
47     printf("usage: zmqsend [OPTIONS]\n");
48     printf("\n"
49           "Options:\n"
50           "-b ADDRESS      set bind address\n"
51           "-h              print this help\n"
52           "-i INFILE       set INFILE as input file, stdin if omitted\n");
53 }
54
55 int main(int argc, char **argv)
56 {
57     AVBPrint src;
58     char *src_buf, *recv_buf;
59     int c;
60     int recv_buf_size, ret = 0;
61     void *zmq_ctx, *socket;
62     const char *bind_address = "tcp://localhost:5555";
63     const char *infile_name = NULL;
64     FILE *infile = NULL;
65     zmq_msg_t msg;
66
67     while ((c = getopt(argc, argv, "b:hi:")) != -1) {
68         switch (c) {

```

```

69     case 'b':
70         bind_address = optarg;
71         break;
72     case 'h':
73         usage();
74         return 0;
75     case 'i':
76         infilename = optarg;
77         break;
78     case '?':
79         return 1;
80     }
81 }
82
83 if (!infilename || !strcmp(infilename, "-")) {
84     infilename = "stdin";
85     infile = stdin;
86 } else {
87     infile = fopen(infilename, "r");
88 }
89 if (!infile) {
90     av_log(NULL, AV_LOG_ERROR,
91           "Impossible to open input file '%s': %s\n", infilename,
92           strerror(errno));
93     return 1;
94 }
95
96 zmq_ctx = zmq_ctx_new();
97 if (!zmq_ctx) {
98     av_log(NULL, AV_LOG_ERROR,
99           "Could not create ZMQ context: %s\n", zmq_strerror(errno));
100 }
101
102 socket = zmq_socket(zmq_ctx, ZMQ_REQ);
103 if (!socket) {
104     av_log(NULL, AV_LOG_ERROR,
105           "Could not create ZMQ socket: %s\n", zmq_strerror(errno));
106     ret = 1;
107     goto end;
108 }
109
110 if (zmq_connect(socket, bind_address) == -1) {
111     av_log(NULL, AV_LOG_ERROR, "Could not bind ZMQ responder to address '%s':
112 %s\n",
113           bind_address, zmq_strerror(errno));
114     ret = 1;
115     goto end;
116 }
117
118 /* grab the input and store it in src */
119 av_bprint_init(&src, 1, AV_BPRINT_SIZE_UNLIMITED);
120 while ((c = fgetc(infile)) != EOF)
121     av_bprint_chars(&src, c, 1);
122 av_bprint_chars(&src, 0, 1);
123
124 if (!av_bprint_is_complete(&src)) {
125     av_log(NULL, AV_LOG_ERROR, "Could not allocate a buffer for the source
126 string\n");
127     av_bprint_finalize(&src, NULL);
128     ret = 1;
129     goto end;
130 }
131 av_bprint_finalize(&src, &src_buf);
132
133 if (zmq_send(socket, src_buf, strlen(src_buf), 0) == -1) {
134     av_log(NULL, AV_LOG_ERROR, "Could not send message: %s\n",
135           zmq_strerror(errno));
136     ret = 1;
137     goto end;
138 }
139
140 if (zmq_msg_init(&msg) == -1) {
141     av_log(NULL, AV_LOG_ERROR,

```

```
139     "Could not initialize receiving message: %s\n",
zmq_strerror(errno));
140     ret = 1;
141     goto end;
142 }
143
144 if (zmq_msg_recv(&msg, socket, 0) == -1) {
145     av_log(NULL, AV_LOG_ERROR,
146         "Could not receive message: %s\n", zmq_strerror(errno));
147     zmq_msg_close(&msg);
148     ret = 1;
149     goto end;
150 }
151
152 recv_buf_size = zmq_msg_size(&msg) + 1;
153 recv_buf = av_malloc(recv_buf_size);
154 if (!recv_buf) {
155     av_log(NULL, AV_LOG_ERROR,
156         "Could not allocate receiving message buffer\n");
157     zmq_msg_close(&msg);
158     ret = 1;
159     goto end;
160 }
161 memcpy(recv_buf, zmq_msg_data(&msg), recv_buf_size - 1);
162 recv_buf[recv_buf_size-1] = 0;
163 printf("%s\n", recv_buf);
164 zmq_msg_close(&msg);
165 av_free(recv_buf);
166
167 end:
168     zmq_close(socket);
169     zmq_ctx_destroy(zmq_ctx);
170     return ret;
171 }
```