zmqsend.c

Go to the documentation of this file.

```c
/*
 * Copyright (c) 2013 Stefano Sabatini
 *
 * This file is part of FFmpeg.
 *
 * FFmpeg is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * FFmpeg is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with FFmpeg; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include "config.h"

#include <stdio.h>
#include <string.h>
#include <zmq.h>

#include "libavutil/log.h"
#include "libavutil/mem.h"
#include "libavutil/bprint.h"

#if HAVE_UNISTD_H
#include <unistd.h>             /* getopt */
#endif

#if !HAVE_GETOPT
#include "compat/getopt.c"
#endif

/**
 * @file
 * zmq message sender example, meant to be used with the zmq filters
 */

static void usage(void)
{
    printf("send message to ZMQ recipient, to use with the zmq filters\n");
    printf("usage: zmqsend [OPTIONS]\n");
    printf("\n"
           "Options:\n"
           "-b ADDRESS        set bind address\n"
           "-h                print this help\n"
           "-i INFILE         set INFILE as input file, stdin if omitted\n");
}

int main(int argc, char **argv)
{
    AVBPrint src;
    char *src_buf, *recv_buf;
    int c;
    int recv_buf_size, ret = 0;
    void *zmq_ctx, *socket;
    const char *bind_address = "tcp://localhost:5555";
    const char *infilename = NULL;
    FILE *infile = NULL;
    zmq_msg_t msg;

    while ((c = getopt(argc, argv, "b:hi:")) != -1) {
        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,
    strerror(errno));
 92          return 1;
 93      }
 94
 95      zmq_ctx = zmq_ctx_new();
 96      if (!zmq_ctx) {
 97          av_log(NULL, AV_LOG_ERROR,
 98                 "Could not create ZMQ context: %s\n", zmq_strerror(errno));
 99          return 1;
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':
    %s\n",
112                 bind_address, zmq_strerror(errno));
113          ret = 1;
114          goto end;
115      }
116
117      /* grab the input and store it in src */
118      av_bprint_init(&src, 1, AV_BPRINT_SIZE_UNLIMITED);
119      while ((c = fgetc(infile)) != EOF)
120          av_bprint_chars(&src, c, 1);
121      av_bprint_chars(&src, 0, 1);
122
123      if (!av_bprint_is_complete(&src)) {
124          av_log(NULL, AV_LOG_ERROR, "Could not allocate a buffer for the source
    string\n");
125          av_bprint_finalize(&src, NULL);
126          ret = 1;
127          goto end;
128      }
129      av_bprint_finalize(&src, &src_buf);
130
131      if (zmq_send(socket, src_buf, strlen(src_buf), 0) == -1) {
132          av_log(NULL, AV_LOG_ERROR, "Could not send message: %s\n",
    zmq_strerror(errno));
133          ret = 1;
134          goto end;
135      }
136
137      if (zmq_msg_init(&msg) == -1) {
138          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 }
```