```c
/* A simple TCP client */
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h> //Added string library
#include <strings.h>//For bzero function
#include <stdlib.h> //Added standard library
#include <unistd.h>
#include <signal.h>

#include "media_transfer.h"
#include "parser.h"

#define SERVER_TCP_PORT     (3000)

int main(int argc, char **argv)
{
    sigaction(SIGPIPE, &(struct sigaction){SIG_IGN}, NULL);

    int     n, bytes_to_read;
    int     batch_mode = 0;
    int     sd, port;
    struct  hostent     *hp;
    struct  sockaddr_in     server;
    char    *host, *bp, rbuf[BUFLEN], sbuf[BUFLEN];

    switch(argc) {
    case 2:
        host = argv[1];
        if (strrchr(host, ':')) {
            port = atoi(strrchr(host, ':') + 1);
            char *ope = strrchr(host, ':');
            *ope = 0;
        } else port = SERVER_TCP_PORT;
        break;
    case 3:
        host = argv[1];
        if (strrchr(host, ':')) {
            port = atoi(strrchr(host, ':') + 1);
            char *ope = strrchr(host, ':');
            *ope = 0;
        } else port = SERVER_TCP_PORT;
        batch_mode = 1;
        break;
    default:
        fprintf(stderr, "Usage: %s <host>[:port] [script]\n", argv[0]);
        exit(1);
    }

    /* Create a stream socket */
    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        fprintf(stderr, "Can't create a socket\n");
        exit(1);
    }

    /* Find the server to connect to */
    bzero((char *)&server, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_port = htons(port);
    if ((hp = gethostbyname(host)) == NULL) {
        fprintf(stderr, "Can't get server's address\n");
        exit(1);
    }

    printf("h_length = %d\n", hp->h_length);

    bcopy(hp->h_addr_list[0], (char *)&server.sin_addr, hp->h_length);
```

```
70
71      /* Connecting to the server */
72      if (connect(sd, (struct sockaddr *)&server, sizeof(server)) == -1) {
73          fprintf(stderr, "Can't connect\n");
74          exit(1);
75      }
76      printf("Connected: server's address is %s\n", hp->h_name);
77
78      if(batch_mode) {
79          process_batch(sd, argv[2]);
80      }
81      else {
82          char time_stamp[TIME_BUFFER_LEN];
83          get_time_spec_to_string(time_stamp, TIME_BUFFER_LEN);
84          while (1) {
85              printf("%s: TX: ", time_stamp);
86              fgets(sbuf, BUFLEN, stdin);          /* get user's text */
87              if(strcmp(sbuf, "exit\n") == 0) {
88                  write(sd, sbuf, BUFLEN);
89                  close(sd);
90                  break;
91              }
92              else {
93                  printf("%s: Sent Command: %s\n", time_stamp, sbuf);
94                  handle_command(sd, sbuf, BUFLEN);
95              }
96          }
97      }
98      return 0;
99  }
```