```c
/*
*   This program is free software: you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*    the Free Software Foundation, either version 3 of the License, or
*    (at your option) any later version.
*
*    This program 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 General Public License for more details.
*
*    You should have received a copy of the GNU General Public License
*    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <time.h>
#include <fcntl.h>

#define BUFF_LEN 256
#define SRVR_ADDR "127.0.0.1"
#define PORT 9000

static void error_hndlr(const char *get) {
    fputs(strerror(errno), stderr);
    fputs(": ", stderr);
    fputs(get, stderr);
    fputs("\n", stderr);

    exit(EXIT_FAILURE);
}

int main() {

    int z; // temp return value
    int is_data; // is there any data?
    int client_socket; // client socket descriptor
    size_t len;
    int fd;

    struct sockaddr_in addr; // socket address
    struct timeval time; // timeval for select()

    fd_set master_fds; // fd sets
    fd_set other_fds;

    char buf[BUFF_LEN];
    char msg[BUFF_LEN];
    char name[BUFF_LEN];

    memset(&buf, 0, sizeof buf); // zero out
    memset(&msg, 0, sizeof msg);

    // get name
    do {
        printf("Name:");
        z = scanf("%s", name);
    } while (z != 1);

    // creating socket
```

```
68      client_socket = socket(AF_INET, SOCK_STREAM, 0);
69      if (client_socket == -1)
70          error_hndlr("Could not open socket()");
71
72      // address
73      memset(&addr, 0, sizeof addr);
74      addr.sin_family = AF_INET;
75      addr.sin_addr.s_addr = inet_addr(SRVR_ADDR);
76      addr.sin_port = htons(PORT);
77      len = sizeof addr;
78
79      // creating connection
80      z = connect(client_socket, (struct sockaddr *) &addr, len);
81      if (z == -1)
82          error_hndlr("Could not connect()");
83
84      fd = fileno(stdin); // fd = standard input descr.
85
86      FD_ZERO(&master_fds);
87      FD_SET(client_socket, &master_fds);
88
89      time.tv_sec = 0;
90      time.tv_usec = 1000;
91
92      for(;;) {
93
94          FD_ZERO(&other_fds);
95          FD_SET(fd, &other_fds);
96
97          if (select(fd + 1, &other_fds, 0, 0, &time)) {
98              fgets(buf, BUFF_LEN, stdin);
99
100             snprintf(msg, sizeof msg, "%s: %s", name, buf);
101
102             z = send(client_socket, msg, sizeof msg, 0);
103             if (z == -1)
104                 error_hndlr("Could not send()");
105
106         }
107
108         ioctl(client_socket, FIONREAD, &is_data); // is there anything to
    read?
109         if (is_data != 0) {
110
111             if (FD_ISSET(client_socket, &master_fds)) {
112                 z = recv(client_socket, buf, sizeof buf, 0);
113                 if (z == -1)
114                     error_hndlr("Could not recv()");
115
116                 printf("%s", buf);
117             }
118         }
119
120     }
121
122     close(client_socket);
123
124     return 0;
125 }
126
127
```