

```

1 // GROUP B
2 // Nathan Baker
3 // nathan.t.baker@okstate.edu
4
5 #include "header.h"
6
7 int connect_to_server(char* ip_addr, int port) {
8     // standard procedure for connecting to a server via TCP socket.
9     int sock = 0;
10    struct sockaddr_in serv_addr;
11    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
12        printf("\nsocket creation error.\n");
13        return -1; // error
14    }
15    serv_addr.sin_family = AF_INET;
16    serv_addr.sin_port = htons(port);
17    if(inet_pton(AF_INET, ip_addr, &serv_addr.sin_addr)<=0) {
18        printf("\ninvalid address.\n");
19        return -1; // error
20    }
21    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
22        printf("\nconnection failed.\n");
23        return -1; // error
24    }
25    return sock; // socket descriptor
26 }
27
28 int main(int argc, char const *argv[]) {
29     int port = 0;
30     if (argc > 1) port = atoi(argv[1]);
31     if (port > 3 || port < 1) { // only servers 1, 2, 3 exist
32         printf("no valid server index provided.\n");
33         exit(1);
34     }
35     char ip_addr[15];
36     strcpy(ip_addr, "127.0.0.1"); // LOCALHOST
37     // strcpy(ip_addr, "10.203.72.25"); // CSX1
38
39     int sock = connect_to_server(ip_addr, 8000+port);
40     if (sock == -1) exit(1);
41     char m[1000]; // to hold messages sent and received
42
43     read(sock, &m, sizeof(m)); // read initial message from server
44     char a = m[0];
45     memmove(m, m+1, 1000); // remove first character and store as flag
46     printf("%s", m);
47     if (a == '2') exit(0); // if flag is 2, server is too busy.
48
49     read(sock, &m, sizeof(m)); // read message from server thread.
50     a = m[0];
51     memmove(m, m+1, 1000); // remove first character and store as flag
52     printf("%s", m);
53     if (a == '2') exit(0); // if flag is 2, end connection.
54
55     while (1) {
56         strcpy(m, ""); // clear message buffer for scanning.
57         if (a != '1') { // if flag is not 1, server is expecting a response.
58             while (strlen(m) == 0) { // ensure customer entered at least 1 character.
59                 scanf("%1000[^\n]", m); // scan until newline.
60                 char z;
61                 if (strlen(m) == 0) scanf("%c", &z);
62             }
63             while ((a = getchar()) != '\n' && a != EOF) { } // flush input steam.
64             send(sock, &m, sizeof(m), 0); // send message.
65         }
66
67         read(sock, &m, sizeof(m));

```

```
68     a = m[0];
69     memmove(m, m+1, 1000); // read message and separate first character flag.
70     printf("%s",m);
71     if (a == '2') exit(0); // if flag is 2, end connection. otherwise continue.
72 }
73
74 return 0;
75 }
```