```
1  //*************************************************************************
2  // ip_server.cc TCP/IP socket server
3  // Author:  M. Thaler
4  // Date:  7.7.99
5  // Changes:     tha, 5/2008
6  //              using getaddrinfo -> IPV4 and IPV6 compatible
7  //*************************************************************************
8
9  //*************************************************************************
10 // system includes
11 //*************************************************************************
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <netinet/in.h>
17 #include <sys/types.h>
18 #include <sys/socket.h>
19 #include <sys/wait.h>
20 #include <unistd.h>
21 #include <sys/un.h>
22 #include <netdb.h>
23
24 #include <iostream>
25 using namespace std;
26
27 //*************************************************************************
28 // local includes
29 //*************************************************************************
30
31 #include "lsocks.h"       // PORT_NUMBER, BUF_SIZE
32
33 //*************************************************************************
34 // Function: main(), parameter: none
35 //*************************************************************************
36
37 int  main(void) {
38
39     int  sfd, cfd, sysRet, j, addrlen, anz;
40     char stringPort[8], buf[BUF_SIZE];;
41     struct addrinfo hints, *aiList, *aiPtr = NULL;
42   struct sockaddr addr;
43
44   cout << endl << "IP server" << endl;
45
46     sprintf(stringPort, "%d", PORT_NUMBER);     // portnumber to string
47     memset(&hints, '\0', sizeof(hints));
48     hints.ai_family   = AF_UNSPEC;
49     hints.ai_socktype = SOCK_STREAM;
50
51     sysRet = getaddrinfo(NULL, stringPort, &hints, &aiList);
52     if (sysRet != 0) {
53         printf("error getting network address %s\n", gai_strerror(sysRet));
54         return(-1);
55     }
56
57     aiPtr = aiList;                             // search through list
58     while (aiPtr != 0) {
59         sfd = socket(aiPtr->ai_family, aiPtr->ai_socktype, aiPtr->ai_protocol);
60         if (sfd >= 0) {
61           j = 1;
62           sysRet = setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR, &j, sizeof(j));
```

```
63            if (sysRet < 0)
64              perror("cannot set socket options");
65
66            if (bind(sfd, aiPtr->ai_addr, aiPtr->ai_addrlen) < 0) {
67              perror("bind failed ");
68              close(sfd);
69              exit(-1);
70            }
71            cout << "Binding successful" << endl;
72            cout << "Connected to port #" << stringPort << endl;
73
74              if (listen(sfd, 5) < 0) {
75                  close(sfd);
76                  perror("listen failed ");
77                  exit(-1);
78              }
79              else
80                  break;
81           }
82           aiPtr = aiPtr->ai_next;
83      }
84      freeaddrinfo(aiList);
85      if (aiPtr == NULL) {
86          printf("could not set up a socket server %s\n");
87          exit(-1);
88      }
89
90    cout << "Listen successful" << endl;
91    cout << "Wainting for client" << endl << endl;
92
93    while ((cfd = accept(sfd, &addr,(unsigned *) &addrlen)) >= 0)  {
94      cout << "Contacted by client" << endl;
95      // get data from client and dispaly on stdout
96      cout << "-> ";
97      while ( (anz = read(cfd, buf, BUF_SIZE)) > 0) {
98        if (buf[0] == '!') break;                      // terminate on '!'
99        if (buf[0] == '@') break;                        // terminate on '@'
100       for (j = 0; j < anz; j++) {
101         cout << buf[j];
102         if (buf[j] == '\n')
103           cout << "-> ";
104       }
105       fflush(stdout);
106     }
107     cout << "Client disconnects" << endl;
108     close(cfd);
109     if (buf[0] == '@')
110             break;                                          // exit on '@'
111     cout << "Waiting for next client" << endl << endl;
112   }
113   cout << "Received '@' from client: good bye" << endl;
114   close(sfd);
115 }
```