```
1  // Demonstrate the use of "select" to implement a multiple socket manager
2  // George F. Riley, Georgia Tech, Spring 2010
3
4  // This is the server side
5
6  #include <iostream>
7  #include <vector>
8
9  #include <sys/socket.h>
10 #include <netdb.h>
11 #include <netinet/in.h>
12 #include <arpa/inet.h>
13 #include <unistd.h>
14
15 using namespace std;
16
17 vector<int> clientSockets;  // Maintains a collection of client socket numbers
18
19 int main(int argc, char** argv)
20 {
21   struct sockaddr_in sockAddr;
22   sockAddr.sin_family = AF_INET;
23   sockAddr.sin_addr.s_addr = htonl(INADDR_ANY);
24   unsigned short port = 2000; // Arbitrarily chosen port number
25   if (argc > 1) port = atol(argv[1]); // Port from command line
26   sockAddr.sin_port = htons(port);
27
28   // Create socket and bind
29   int listenSock = socket(AF_INET, SOCK_STREAM, 0);
30   if (bind(listenSock, (struct sockaddr*)&sockAddr, sizeof(sockAddr)) < 0)
31     {
32       std::cout << "Bind failed" << std::endl;
33       exit(1);
34     }
35   listen(listenSock, 5);
36
37   while(true)
38     {
39       fd_set readSet;
40       fd_set writeSet;
41       fd_set errorSet;
42       // Clear all bits to zero
43       FD_ZERO(&readSet);
44       FD_ZERO(&writeSet);
45       FD_ZERO(&errorSet);
46       // Now set the read bit for the listening socket
47       int maxSock = listenSock;
48       FD_SET(listenSock, &readSet);
49       // Set the read bits for each client socket
50       for (unsigned i = 0; i < clientSockets.size(); ++i)
51         {
52           if (clientSockets[i] < 0) continue; // No longer used
53           if (clientSockets[i] > maxSock) maxSock = clientSockets[i];
54           FD_SET(clientSockets[i], &readSet);
55         }
56       struct timeval t; // This specifies wait time
```

Program chatserv.cc

```
57        t.tv_sec = 1;     // Wait for one second
58        t.tv_usec = 0;
59        select(maxSock + 1, &readSet, &writeSet, &errorSet, &t);
60        // See if listening socket can be "read", ie. call accept
61        if (FD_ISSET(listenSock, &readSet))
62          {
63            int s = accept(listenSock, 0, 0);
64            cout << "Got accept, socket " << s << endl;
65            clientSockets.push_back(s);
66          }
67        // Check each client for input data
68        char buf[10000];
69        for (unsigned i = 0; i < clientSockets.size(); ++i)
70          {
71            if (clientSockets[i] < 0) continue; // No longer used
72            if (FD_ISSET(clientSockets[i], &readSet) ||
73                FD_ISSET(clientSockets[i], &errorSet))
74              {
75                //cout << "Reading socket " << clientSockets[i] << endl;
76                int actual = read(clientSockets[i], buf, sizeof(buf));
77                //cout << "Got " << actual << " bytes" << endl;
78                if (actual <= 0)
79                  { // Closed or error
80                    cout << "Remote closed socket " << clientSockets[i]
81                         << endl;
82                    clientSockets[i] = -1; // Note no longer used
83                  }
84                else
85                  {
86                    // And echo this data to each client
87                    for (unsigned j = 0; j < clientSockets.size(); ++j)
88                      {
89                        if (clientSockets[j] < 0) continue;
90                        int wrote = write(clientSockets[j], buf, actual);
91                        //cout << "Wrote " << wrote << " bytes to socket "
92                        //     <<  clientSockets[i] << endl;
93                      }
94                  }
95              }
96          }
97      }
98  }
```

Program chatserv.cc (continued)

```
1  // Demonstrate the use of "select" to implement a multiple socket manager
2  // George F. Riley, Georgia Tech, Spring 2010
3
4  // This is the client side
5
6  #include <stdio.h>
7  #include <iostream>
8  #include <vector>
9
10 #include <sys/socket.h>
11 #include <netdb.h>
12 #include <netinet/in.h>
13 #include <arpa/inet.h>
14 #include <unistd.h>
15
16 using namespace std;
17
18 int main(int argc, char** argv)
19 {
20   if (argc < 2)
21     {
22       cout << "Usage: chat hostname" << endl;
23       exit(1);
24     }
25
26   int s = socket(AF_INET, SOCK_STREAM, 0);
27   struct hostent* pHE = gethostbyname(argv[1]);
28   if (pHE == 0)
29     { // not found
30       cout << "Can't find IP address for host " << argv[1] << endl;
31       exit(2);
32     }
33
34   struct sockaddr_in sockAddr;
35   sockAddr.sin_family = AF_INET;
36   // Get the ip address from the hostent structure
37   memcpy(&sockAddr.sin_addr, pHE->h_addr, 4);
38   unsigned short port = 2000; // Arbitrarily chosen port number
39   if (argc > 2) port = atol(argv[2]);
40   sockAddr.sin_port = htons(port);
41
42   // Create socket and bind
43   if (connect(s, (struct sockaddr*)&sockAddr, sizeof(sockAddr)) < 0)
44     {
45       std::cout << "connect failed" << std::endl;
46       exit(1);
47     }
48   while(true)
49     {
50       // We need to use select to read either from standard in (handle = 1)
51       // or the socket
52       fd_set readSet;
53       fd_set errorSet;
54       FD_ZERO(&readSet);
55       FD_ZERO(&errorSet);
56       // Set two bits in the readSet bitmap, standard in and the socket
```

Program chat.cc

```
57        FD_SET(STDIN_FILENO, &readSet);
58        FD_SET(STDIN_FILENO, &errorSet);
59        FD_SET(s, &readSet);
60        FD_SET(s, &errorSet);
61        // here we don't use the write set or the time outvalue
62        select(s + 1, &readSet, 0, &errorSet, 0);
63        if (FD_ISSET(s, &readSet) ||
64            FD_ISSET(s, &errorSet))
65          { // Read from socket
66            char buf[10000];
67            int actual = read(s, buf, sizeof(buf));
68            if (actual <= 0) break; // Remote closed connection
69            // write the data to standard out
70            write(STDOUT_FILENO, buf, actual);
71          }
72          if (FD_ISSET(STDIN_FILENO, &readSet) ||
73              FD_ISSET(STDIN_FILENO, &errorSet))
74          { // Data available from keyboard
75            char buf[10000];
76            int actual = read(STDIN_FILENO, buf, sizeof(buf));
77            if (actual <= 0) break;  // End of file
78            // Write to socket
79            int wrote = write(s, buf, actual);
80          }
81      }
82    close(s);
83  }
```

Program chat.cc (continued)