Lesson 3 - Socket Programming
Building a Simple Client

Summary

We are going to use the functions described in the first lesson to build a very simple client program to message our server program we created in the second tutorial.

The Client Code

The code for creating our client is very similar to that of the server. We still need to initialise the sockets library and creating our addressing information in the same way as the server. However, when creating our addressing structs we do not need to bind to the socket so we do not need to pass the `AI_PASSIVE` flag to the hints struct.

```
1  int main(void)
2  {
3     SOCKET s = NULL;
4
5     // initialise socket library
6     if (init())
7     {
8        printf("Unable to initialise the Winsock library\n");
9        exit(1);
10    }
11
12    // intiailise addressing information
13    if (addressing() != 0)
14    {
15       printf("Uanble to initialise addressing information\n");
16       exit(1);
17    }
18
19    // create a socket for the server to listen on
20    if ((s = socket(addr->ai_family, addr->ai_socktype,
21       addr->ai_protocol)) == INVALID_SOCKET)
22    {
23       printf("Unable to create a socket\n");
24       printf("Failed with error: %d\n%s\n", WSAGetLastError(),
25          gai_strerror(WSAGetLastError()));
26       exit(1);
27    }
28    else
29    {
30       printf("\nSocket created successfully.\n");
31    }
32
33    // connect to the server
34    if (connect(s, addr->ai_addr, addr->ai_addrlen) != 0)
35    {
```

```
36          printf("Unable to connect to server\n");
37          printf("Failed with error: %d\n%s\n", WSAGetLastError(),
38              gai_strerror(WSAGetLastError()));
39          //exit(1);
40      }
41      else
42      {
43          printf("\nConnected to the server.\n");
44      }
45
46      // finished with addrinfo struct now
47      freeaddrinfo(addr);
48
49      // accept message from server and close
50      int bytesreceived;
51      char buff[BUFFSIZE];
52
53      if ((bytesreceived = recv(s, buff, BUFFSIZE-1, 0)) == -1)
54      {
55          printf("Error receiving\n");
56          printf("Failed with error: %d\n%s\n", WSAGetLastError(),
57              gai_strerror(WSAGetLastError()));
58      }
59      else
60      {
61          buff[bytesreceived] = '\0';
62          printf("Message received. Received %d bytes.\nMessage is: %s",
63              bytesreceived,  buff);
64      }
65
66      char* hw = "Hello Server";
67      send(s, hw, strlen(hw), 0);
68
69      closesocket(s);
70      WSACleanup();
71
72      while(1);
73      //exit(0);
74  }
```

Client program

Where the server calls `bind()` on the socket that will accept incomming connections, a client program must instead `connect()` to that socket. Once connected, the client can send and receive messages to and from the server.