# C Program for Server Side

1. int portno=5000; //can use any between 1024 and 65535
2. struct sockaddr_in serv_addr; /* struct sockaddr_in{ short sin_family; //it is an address family that is used to designate the type of addresses that socket can communicate with (in this case, IPv4, IPv6 addresses). For the most part, sticking with AF_INET for socket programming over a network is the safest option.
   unsigned short sin_port; // port number to communicate with struct in_addr sin_addr; // server IP address

char sin_zero[8]; // padding zeros to make structure same size as SOCKADDR.
   };
*/
//<netinet/in.h>
3. bzero((char *) &serv_addr, sizeof(serv_addr)); //bzero initializes structure with zero
4. serv_addr.sin_family = AF_INET; //used for IPv4
5. serv_addr.sin_addr.s_addr = INADDR_ANY; // if specified, socket is bind to any available local IP address 0.0.0.0
6. serv_addr.sin_port = htons(portno); //specifies port number to bind with, htons converts short integer to network byte order
7. int sockfd; // used for storing socket descriptor, this only listens to any of the client's connect request
8. sockfd = socket(AF_INET, SOCK_STREAM, 0);

// AF_INET for IPv4, SOCK_STREAM is for creating TCP connection, SOCK_DGRAM is for creating UDP connection, 0 is specified TCP/UDP protocol, otherwise for RAW_STREAM valid IANA protocol needs to be specified.

9. if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0 ) { error("ERROR on binding"); }

//bind IP address and port number to create a socket

10. listen(sockfd,5);

/*

first argument specifies socket descriptor where information from client will be stored

Second argument defines the maximum length to which the queue of pending connections for *sockfd* may grow.

*/

}

11.struct sockaddr_in cli_addr; //storing client address

12.socklen_t clilen; //storing length for client address, i.e. 32 bit integer

13.clilen = sizeof(cli_addr);

14.int newsockfd; //socket descriptor for client, this is exclusively returned for the specific client

15.newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen); //accept returns a socket descriptor through which client and server communicate

16.char buffer[256]; int n; // buffer for storing client information

17.n = read(newsockfd,buffer,255); //reads information from socket to local buffer

18.printf("Here is the message: %s\n", buffer);

19.n = write(newsockfd,"I got your message",18); //writes message to the socket descriptor 20. close(newsockfd);

21. close(sockfd);

# C Program for Client Side

1. int sockfd, portno=5000, n;
2. sockfd = socket(AF_INET, SOCK_STREAM, 0);
3. struct hostent *server; //<netdb.h>

4. server = gethostbyname("localhost"); //this command resolves host address to corresponding IP address
5. struct sockaddr_in serv_addr;
6. bzero((char *) &serv_addr, sizeof(serv_addr)); // initializes buffer
7. serv_addr.sin_family = AF_INET; // for IPv4 family
8. bcopy((char *)server->h_addr, (char *) // copy server IP address

&serv_addr.sin_addr.s_addr, server->h_length);
9. serv_addr.sin_port = htons(portno); //defining port number
10. if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0 ) { error("ERROR connecting"); } // initiating connect request to the server
11. char buffer[256]; int n,m; // client buffer to forward request to the server
12. getline(&buffer, &m, stdin);
13. n = write(sockfd, buffer, strlen(buffer));
14. bzero(buffer,255);
15. n = read(sockfd,buffer,255);
16. printf("%s\n", buffer);
17. close(sockfd);