

```

1  SERVER TCP
2
3  /*This server program is concurrent*/
4  #include <sys/types.h>
5  #include <sys/socket.h>
6  #include <stdio.h>
7  #include <netinet/in.h>
8  #define MAXLINE_SIZE 100
9  #define SERV_PORT 5555
10 int listensd, clientsd;
11 char buffer[MAXLINE_SIZE+1];
12 struct sockaddr_in servaddr;
13 int noBytesRead=0;
14
15
16 /*this function will server client that connects*/
17 void processClient(int);
18
19 int main()
20 {
21     /*Create socket*/
22     if((listensd=socket(AF_INET, SOCK_STREAM, 0))<0)
23     {
24         fprintf(stderr, "Cannot create socket\n");
25         exit(-1);
26     }
27
28     /*Initialize socket address structure*/
29     bzero(&servaddr, sizeof(servaddr));
30     servaddr.sin_family=AF_INET;
31     servaddr.sin_port=htons(SERV_PORT);
32     servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
33
34     /*bind socket address to the socket*/
35     if(bind(listensd, (struct sockaddr*)&servaddr, sizeof(servaddr))<0)
36     {
37         fprintf(stderr, "Error in bind\n");
38         exit(-1);
39     }
40
41     /*Make the socket listening socket*/
42     if(listen(listensd, 5)<0)
43     {
44         fprintf(stderr, "Error in listen\n");
45         exit(-1);
46     }
47
48     for(;;)
49     {
50         /*wait for client connection*/
51         clientsd=accept(listensd, (struct sockaddr*)NULL, NULL);
52         if(fork()==0)
53         {
54             /*close listening socket in child.
55             So that reference count remains one.
56             The child serves the client.
57             It does not need listening socket to do this. */
58             close(listensd);
59
60             /*server client*/
61             processClient(clientsd);
62             /*close connected socket*/
63             close(clientsd);
64             exit(0);
65         }
66         /*close connected socket in parent so that reference count remains one. */
67         close(clientsd);
68     }
69     return 0;
70 }
71
72 void processClient(int clientsd)
73 {

```

```

74     /*read message from client and send back*/
75     while((noBytesRead=read(clientsd,buffer,sizeof(buffer)))>0)
76         write(clientsd,buffer,noBytesRead);
77 }
78
79
80
81 Client
82
83 #include <sys/types.h>
84 #include <sys/socket.h>
85 #include <stdio.h>
86 #include <netinet/in.h>
87 #include <string.h>
88 #define MAXLINESIZE 100
89 #define SERV_PORT 5555
90
91 int main(int argc,char** argv)
92 {
93     int connectsd;
94     char sendBuffer[MAXLINESIZE+1];
95     char recvBuffer[MAXLINESIZE+1];
96     struct sockaddr_in servaddr;
97     int noBytesRead=0;
98
99     if(argc!=2)
100     {
101         fprintf(stderr,"Usage: %s IP-Address\n",argv[0]);
102         exit(-1);
103     }
104
105     /*Create socket*/
106     if((connectsd=socket(AF_INET,SOCK_STREAM,0))<0)
107     {
108         fprintf(stderr,"Cannot create socket\n");
109         exit(-1);
110     }
111
112     /*Initialize socket address structure*/
113     bzero(&servaddr,sizeof(servaddr));
114     servaddr.sin_family=AF_INET;
115     servaddr.sin_port=htons(SERV_PORT);
116
117     /*assign server address in socket address structure*/
118     if(inet_pton(PF_INET,argv[1],&servaddr.sin_addr)<=0)
119     {
120         fprintf(stderr,"Error in inet_pton\n");
121         exit(-1);
122     }
123
124     /*Get connected with the server*/
125     if(connect(connectsd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
126     {
127         fprintf(stderr,"Error in connect\n");
128         exit(-1);
129     }
130
131     /*Read message from user through keyboard*/
132     for(;gets(sendBuffer)!=NULL;)
133     {
134         /*Send the message to the server*/
135         write(connectsd,sendBuffer,strlen(sendBuffer)+1);
136         if(noBytesRead=read(connectsd,recvBuffer,sizeof(recvBuffer))<0)
137             exit(0);
138
139         /*Display what the server sent in reply*/
140         fprintf(stdout,"%s\n",recvBuffer);
141     }
142     return 0;
143 }
144

```