/* tcpserver.c */

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main()
{
    int sock, connected, bytes_recieved , true = 1;
    char send_data [1024] , recv_data[1024];

    struct sockaddr_in server_addr,client_addr;
    int sin_size;

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
       perror("Socket");
       exit(1);
    }

    if (setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&true,sizeof(int)) == -1) {
       perror("Setsockopt");
       exit(1);

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(5000);
```

```
server_addr.sin_addr.s_addr = INADDR_ANY;
bzero(&(server_addr.sin_zero),8);

if (bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr))
                                          == -1) {
   perror("Unable to bind");
   exit(1);
}

if (listen(sock, 5) == -1) {
   perror("Listen");
   exit(1);
}

     printf("\nTCPServer Waiting for client on port 5000");
fflush(stdout);



while(1)
{

   sin_size = sizeof(struct sockaddr_in);

   connected = accept(sock, (struct sockaddr *)&client_addr,&sin_size);

   printf("\n I got a connection from (%s , %d)",
        inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));

   while (1)
   {
    printf("\n SEND (q or Q to quit) : ");
```

```
            gets(send_data);

            if (strcmp(send_data , "q") == 0 || strcmp(send_data , "Q") == 0)
            {
              send(connected, send_data,strlen(send_data), 0);
              close(connected);
              break;
            }

            else
               send(connected, send_data,strlen(send_data), 0);

            bytes_recieved = recv(connected,recv_data,1024,0);

            recv_data[bytes_recieved] = '\0';

            if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
            {
              close(connected);
              break;
            }

            else
            printf("\n RECIEVED DATA = %s " , recv_data);
            fflush(stdout);
          }
       }

    close(sock);
    return 0;
}
```

/* tcpclient.c */

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

int main()
{
      int sock, bytes_recieved;
      char send_data[1024],recv_data[1024];
      struct hostent *host;
      struct sockaddr_in server_addr;

      host = gethostbyname("10.10.12.102");

      if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
         perror("Socket");
         exit(1);
      }
      server_addr.sin_family = AF_INET;
      server_addr.sin_port = htons(5002);
      server_addr.sin_addr = *((struct in_addr *)host->h_addr);
      bzero(&(server_addr.sin_zero),8);

      if (connect(sock, (struct sockaddr *)&server_addr,
```

```
            sizeof(struct sockaddr)) == -1)
    {
       perror("Connect");
       exit(1);
    }
    while(1)
    {
     bytes_recieved=recv(sock,recv_data,1024,0);
     recv_data[bytes_recieved] = '\0';

     if (strcmp(recv_data , "q") == 0 || strcmp(recv_data , "Q") == 0)
     {
      close(sock);
      break;
     }
     else
      printf("\nRecieved data = %s " , recv_data);
      printf("\nSEND (q or Q to quit) : ");
      gets(send_data);

     if (strcmp(send_data , "q") != 0 && strcmp(send_data , "Q") != 0)
      send(sock,send_data,strlen(send_data), 0);
     else
     {
      send(sock,send_data,strlen(send_data), 0);
      close(sock);
      break;
     }
    }
return 0;
}
```

/* udpserver.c */

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    int sock;
    int addr_len, bytes_read;
    char recv_data[1024];
    struct sockaddr_in server_addr , client_addr;


    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       perror("Socket");
       exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(5000);
    server_addr.sin_addr.s_addr = INADDR_ANY;
    bzero(&(server_addr.sin_zero),8);
```

```
    if (bind(sock,(struct sockaddr *)&server_addr,
       sizeof(struct sockaddr)) == -1)
    {
       perror("Bind");
       exit(1);
    }

    addr_len = sizeof(struct sockaddr);

        printf("\nUDPServer Waiting for client on port 5000");
    fflush(stdout);

        while (1)
        {

     bytes_read = recvfrom(sock,recv_data,1024,0,
                        (struct sockaddr *)&client_addr, &addr_len);


          recv_data[bytes_read] = '\0';

     printf("\n(%s , %d) said : ",inet_ntoa(client_addr.sin_addr),
                              ntohs(client_addr.sin_port));
     printf("%s", recv_data);
          fflush(stdout);

    }
    return 0;
}
```

/* udpclient.c */

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>

int main()
{
int sock;
struct sockaddr_in server_addr;
struct hostent *host;
char send_data[1024];

host= (struct hostent *) gethostbyname((char *)"127.0.0.1");



if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
{
perror("socket");
exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(5000);
```

```
server_addr.sin_addr = *((struct in_addr *)host->h_addr);
bzero(&(server_addr.sin_zero),8);

  while (1)
  {

   printf("Type Something (q or Q to quit):");
   gets(send_data);

   if ((strcmp(send_data , "q") == 0) || strcmp(send_data , "Q") == 0)
     break;

   else
     sendto(sock, send_data, strlen(send_data), 0,
          (struct sockaddr *)&server_addr, sizeof(struct sockaddr));

  }

}
```