```c
/*****************************************************************************
 *** Program explaining the client-server model                            ***
 *** This is the client program                                            ***
 *** developed by Ashok Kumar Das, CSE Department, IIT Kharagpur           ***
 ***                                                                       ***
 *****************************************************************************/


#include <stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>

/* Global constants */
#define SERVICE_PORT 41041
#define MAX_LEN 1024

#define DEFAULT_SERVER "192.168.1.241"

#define REQ 10  /* A request message */
#define ACK 20  /* An acknowledgement */

/* Define the header of a message structure */
typedef struct {
 int opcode;
 int src_addr;
 int dest_addr;
 } Hdr;


/* Define the body of a message */
typedef struct {
 Hdr hdr;
 char buf[MAX_LEN];
 int a[MAX_LEN]; /* contains the MAX_LEN integers */
 int n; /* Number of elements in the array a[] */
} Msg;


/* Function prototypes */
int serverConnect ( char * );
void Talk_to_server ( int );

/* Connect with the server: socket() and connect() */
int serverConnect ( char *sip )
{
   int cfd;
   struct sockaddr_in saddr;   /* address of server */
   int status;

   /* request for a socket descriptor */
   cfd = socket (AF_INET, SOCK_STREAM, 0);
   if (cfd == -1) {
      fprintf (stderr, "*** Client error: unable to get socket descriptor\n");
      exit(1);
   }

   /* set server address */
   saddr.sin_family = AF_INET;              /* Default value for most applications */
   saddr.sin_port = htons(SERVICE_PORT);
                                            /* Service port in network byte order */
   saddr.sin_addr.s_addr = inet_addr(sip);  /* Convert server's IP to short int */
   bzero(&(saddr.sin_zero),8);              /* zero the rest of the structure */
```

```
   /* set up connection with the server */
   status = connect(cfd, (struct sockaddr *)&saddr, sizeof(struct sockaddr));
   if (status == -1) {
      fprintf(stderr, "*** Client error: unable to connect to server\n");
      exit(1);
   }

   fprintf(stderr, "Connected to server\n");

   return cfd;
}


/* Interaction with the server */
void Talk_to_server ( int cfd )
{
   char buffer[MAX_LEN];
   int nbytes, status;
   int src_addr, dest_addr;
   int i, n;
   Msg send_msg;
   Msg recv_msg;

   dest_addr = inet_addr("DEFAULT_SERVER");
   src_addr = inet_addr("192.168.1.245");

   /* send the request message to the server */
   printf("Sending the request message to the server \n");
   send_msg.hdr.opcode = REQ;
   send_msg.hdr.src_addr = src_addr;
   send_msg.hdr.dest_addr = dest_addr;
   strcpy(send_msg.buf,"Request message from the client\n");

   printf("Enter the array size, n: ");
   scanf("%d", &n);

   if ( n <= MAX_LEN ) {
     send_msg.n = n;
     printf("The client sends the following %d integers: \n", n);
     /* Read integers in the array randomly inbetween 1 and 1000 */
     for (i= 0; i < n; i++) {
       srand((unsigned int) time(NULL) + rand() % 5000);
       send_msg.a[i] = rand() % 1000 + 1;
       printf("%6d", send_msg.a[i]);
     }
    printf("\n");
   }
   /* send the request message to the server */
   status = send(cfd, &send_msg, sizeof(Msg), 0);
   if (status == -1) {
      fprintf(stderr, "*** Client error: unable to send\n");
      return;
    }

  /* Wait for responses from the server */
   while ( 1 ) {

   /* receive messages from server */
   nbytes = recv(cfd, &recv_msg, sizeof(Msg), 0);
   if (nbytes == -1) {
      fprintf(stderr, "*** Client error: unable to receive\n");

   }
  sleep(10);

  switch ( recv_msg.hdr.opcode ) {

   case REQ : /* Request message */
   printf("REQ message from the server: Received integers are: \n");
        for (i=0; i<recv_msg.n; i++)
```

```
          printf("%6d", recv_msg.a[i]);
        printf("\n");
             /* Send an acknowlegement message to the server */
           printf("Sending the request message to the server \n");
   send_msg.hdr.opcode = ACK;
   send_msg.hdr.src_addr = src_addr;
   send_msg.hdr.dest_addr = dest_addr;
   strcpy(send_msg.buf,"Acknowledgement message from the client\n");
   /* send the request message to the server */
   status = send(cfd, &send_msg, sizeof(Msg), 0);
   if (status == -1) {
      fprintf(stderr, "*** Client error: unable to send\n");
      return;
    }
    break;


  case ACK: /* Acknowlegement message from the server */
        printf("%s\n", recv_msg.buf);
    break;

 default: /* Erroneous message received */
       printf("Invalid message received from the server\n");
     exit(0);
  }
 }
}

int main ( int argc, char *argv[] )
{
   char sip[16];
   int cfd;


   printf("******* This is client side demo program for implementing communication in Client-Server
Systems *****\n\n");
   strcpy(sip, (argc == 2) ? argv[1] : DEFAULT_SERVER);
   cfd = serverConnect(sip);
   Talk_to_server (cfd);
   close(cfd);
}

/*** End of client.c ***/
```