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


#include <stdio.h>
#include <stdlib.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 Q_SIZE 5
#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 startServer ( );
void serverLoop ( int );
void Talk_to_client ( int );


/* Start the server: socket(), bind() and listen() */
int startServer ()
{
   int sfd;                     /* for listening to port PORT_NUMBER */
   struct sockaddr_in saddr;    /* address of server */
   int status;

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

   /* Set the fields of server's internet address structure */
   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 = INADDR_ANY;    /* Server's local address: 0.0.0.0 (htons not necessary) */
```

```
   bzero(&(saddr.sin_zero),8);            /* zero the rest of the structure */

   /* Bind the socket to SERVICE_PORT for listening */
   status = bind(sfd, (struct sockaddr *)&saddr, sizeof(struct sockaddr));
   if (status == -1) {
      fprintf(stderr, "*** Server error: unable to bind to port %d\n", SERVICE_PORT);
      exit(2);
   }

   /* Now listen to the service port */
   status = listen(sfd,Q_SIZE);
   if (status == -1) {
      fprintf(stderr, "*** Server error: unable to listen\n");
      exit(3);
   }

   fprintf(stderr, "+++ Server successfully started, listening to port %hd\n", SERVICE_PORT);
   return sfd;
}


/* Accept connections from clients, spawn a child process for each request */
void serverLoop ( int sfd )
{
   int cfd;                     /* for communication with clients */
   struct sockaddr_in caddr;    /* address of client */
   int size;

    while (1) {
      /* accept connection from clients */
      cfd = accept(sfd, (struct sockaddr *)&caddr, &size);
      if (cfd == -1) {
         fprintf(stderr, "*** Server error: unable to accept request\n");
         continue;
      }
      fprintf(stderr, "**** Connected with %s\n", inet_ntoa(caddr.sin_addr));

 /* fork a child to process request from client */
      if (!fork()) {
         Talk_to_client (cfd);
         fprintf(stderr, "**** Closed connection with %s\n", inet_ntoa(caddr.sin_addr));
         close(cfd);
         exit(0);
      }

      /* parent (server) does not talk with clients */
      close(cfd);

      /* parent waits for termination of child processes */
      while (waitpid(-1,NULL,WNOHANG) > 0);


   }

}

/* Interaction of the child process with the client */
void Talk_to_client ( int cfd )
{
   char buffer[MAX_LEN];
   int nbytes, status;
   int src_addr, dest_addr;
   int i = 0;
   Msg send_msg;
   Msg recv_msg;

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

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

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

   }

  sleep(5);

  switch ( recv_msg.hdr.opcode ) {

  case REQ : /* Request message */

        printf("REQ message from the client: 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 cliet */
   printf("Sending the acknoledgement message to the client \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 server\n");
   /* send the ackowledgement message to the client  */
   status = send(cfd, &send_msg, sizeof(Msg), 0);
   if (status == -1) {
      fprintf(stderr, "*** Server error: unable to send\n");
      return;
    }
    break;


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

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

int main ()
{
   int sfd;

   printf("******* This is the server side demo program for implementing communication in Client-Server
Systems *****\n\n");
   sfd = startServer();


   serverLoop(sfd);
}
/*** End of server.c ***/
```