**Q1. WAP using client-server socket programming: A client enters an integer number n and server returns the results Fi where 0<= i <=n. Fi denotes the fibonacci number of ith position.**

**E.g., if n=6**

**Output: F0 = 0, F1 = 1, F2 = 1, F3 = 2, F4 = 3, F5 = 5, F6 = 8**

**Client:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nClient: - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectSocket(int hSocket)

{

int iRetval = -1;

int serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote,sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataToServer(int hSocket, int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int),0);

return shortRetVal;

}

int main()

{

int hSocket,read\_size,numberToServer=0,serverResponse[100];

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket==-1)

{

printf("\nClient: - Could not create socket");

return 1;

}

printf("\nClient: - Socket created...");

if(connectSocket(hSocket)<0)

{

printf("\nClient: - Connection failed. Exiting...");

return 1;

}

printf("\nClient: - Socket connected successfully...");

printf("\nClient: - Enter n, for upto n fibonacci terms : ");

scanf("%d",&numberToServer);

if(numberToServer<0)

{

printf("\nServer : - Invalid input!!!");

return 0;

}

if(sendDataToServer(hSocket,&numberToServer)<0)

{

printf("\nClient: - Data uplink failed. Exiting...");

return 1;

}

if(recv(hSocket,&serverResponse,(100)\*sizeof(int),0)<0)

{

printf("\nClient: - Data receiving failed");

return 1;

}

printf("\nServer : - ");

for(int i=0; i <= numberToServer; ++i)

printf("F(%d) = %d ",i,serverResponse[i]);

printf("\n");

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nServer: - Creating Socket\n");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindSocket(int hSocket)

{

int iRetval=-1;

int ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

void fibonacci(int num,int ans[])

{

int i;

ans[0]=0;ans[1]=1;

for(i=2;i<=num;++i)

ans[i]=ans[i-1]+ans[i-2];

}

int main()

{

int socket\_desc,client\_sock,c,read\_size,numberFromClient=0,fibonacciNumber[100],opt=1;

struct sockaddr\_in server, client;

socket\_desc=createSocket();

if(socket\_desc == -1)

{

printf("\nServer: - Could not create socket");

return 1;

}

printf("\nServer: - Socket created");

if(bindSocket(socket\_desc)<0)

printf("\nServer: - Bind failed");

printf("\nServer: - Binding done");

listen(socket\_desc,3);

printf("\nServer: - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc, (struct sockaddr \*)&client, (socklen\_t\*)&c);

if(client\_sock<0)

{

printf("\nServer: - Connnection not accepted");

return 1;

}

printf("\nServer: - Connnection accepted");

if(recv(client\_sock, &numberFromClient, 1\*sizeof(int), 0) < 0 )

{

printf("\nServer: - Data receiving Failed\n");

return 0;

}

printf("\nServer: - The number send by client is :%d\n",numberFromClient);

fibonacci(numberFromClient, fibonacciNumber);

if(send(client\_sock, fibonacciNumber, sizeof(fibonacciNumber)/sizeof(int), 0 ) < 0 )

{

printf("\nServer: - Send data to Client failed\n");

return 1;

}

close(client\_sock);

return 0;

}

**Test Run:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc FibC.c -o fc

nitish89@ADMINRG-R5IIR8M ~

$ gcc FibS.c -o fs

nitish89@ADMINRG-R5IIR8M ~

$ ./fC

Client: - Creating Socket

Client: - Socket created...

Client: - Socket connected successfully...

Client: - Enter n, for upto n fibonacci terms : 6

Server : - F(0) = 0 F(1) = 1 F(2) = 1 F(3) = 2 F(4) = 3 F(5) = 5 F(6) = 8

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./fS

Server: - Creating Socket

Server: - Socket created

Server: - Binding done

Server: - Waiting for connections...

Server: - Connnection accepted

Server: - The number send by client is :6

**Q2. WAP using client-server socket programming: Client will enter an integer number n and server will return its factorial.**

**E.g., if n=6**

**Output: a = n! = 720**

**Client:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nClient :- Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket)

{

int iRetval=-1,serverPort=8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendUsingSocket(int hSocket, int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main()

{

int hSocket,read\_size,numberToServer=0,serverResponse=1;

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket == -1)

{

printf("\nClient :- Could not create socket");

return 1;

}

printf("Client :- Socket created\n");

if(connectToSocket(hSocket) < 0 )

{

printf("\nClient :- Connection Failed.");

return 1;

}

printf("\nClient :- Successfully connected to socket");

printf("\nClient :- Enter a number : ");

scanf("%d", &numberToServer);

if(numberToServer < 0 )

{

printf("\nClient :- Invalid input\n");

return 0;

}

if(sendUsingSocket(hSocket, &numberToServer)<0)

{

printf("\nClient :- Data uplink failed\n");

return 1;

}

if(recv(hSocket,&serverResponse,1\*sizeof(int),0)<0)

{

printf("\nClient :- Receiving server response failed\n");

return 1;

}

printf("\nClient : - Received reply from server.");

printf("\nClient : - Factorial of given number as returned by server is : %d",serverResponse);

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nServer : - Creating Socket\n");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket)

{

int iRetval=-1,ClientPort=8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main()

{

int socket\_desc,client\_sock,c,read\_size,numberFromClient=0,factorial=1;

struct sockaddr\_in server, client;

socket\_desc = createSocket();

if(socket\_desc == -1)

{

printf("\nServer : - Could not create socket");

return 1;

}

printf("\nServer : - Socket created");

if(bindCreatedSocket(socket\_desc)<0)

{

printf("\nServer : - Socket binding failed");

return 1;

}

printf("\nServer : - Socket binding done");

listen(socket\_desc,3);

printf("\nServer : - Waiting for connections...");

c=sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc, (struct sockaddr \*)&client, (socklen\_t\*)&c);

if(client\_sock<0)

{

printf("\nServer : - Connection from client rejected.");

return 1;

}

printf("\nServer : - Connnection from client accepted");

if(recv(client\_sock, &numberFromClient, 1\*sizeof(int), 0) < 0 )

{

printf("\nServer : - Data uplink from client failed.");

return 0;

}

printf("\nServer : - The number send by client is :%d",numberFromClient);

for(int i=1; i<=numberFromClient; ++i)

factorial \*= i;

if(send(client\_sock, &factorial, 1 \* sizeof(int), 0 ) < 0 )

{

printf("\nServer : - Data downlink from server failed.");

return 1;

}

close(client\_sock);

return 0;

}

**Test Run:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc FactC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc FactS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client :- Creating Socket.

Client :- Socket created

Client :- Successfully connected to socket

Client :- Enter a number : 6

Client : - Received reply from server.

Client : - Factorial of given number as returned by server is : 720

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket

Server : - Socket created

Server : - Socket binding done

Server : - Waiting for connections...

Server : - Connnection from client accepted

Server : - The number send by client is :6

**Q3. WAP using client-server socket programming: Server keeps grade record as 85-100 : Grade A, 70-84 : Grade B, 60-69 : Grade C, 50-59 : Grade D, 0-50 : fail. Client enters a mark n and get the grade.**

**E.g., if n = 75**

**Output: Grade B**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket)

{

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket,int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main()

{

int hSocket, read\_size,numberToServer=0;

struct sockaddr\_in server;

char serverResponse;

hSocket = createSocket();

if(hSocket == -1)

{

printf("\nClient : - Could not create socket");

return 1;

}

printf("\nClient : - Socket created");

if(connectToSocket(hSocket) < 0 )

{

printf("\nClient : - Connection to socket failed.");

return 1;

}

printf("\nClient : - Successfully connected to socket");

printf("\nClient : - Enter marks obtained by a student : ");

scanf("%d", &numberToServer);

if(numberToServer < 0 )

{

printf("\nClient : - Invalid input");

return 0;

}

if(sendDataUsingSocket(hSocket, &numberToServer) < 0 )

{

printf("\nClient : - Data uplink to server failed.");

return 1;

}

if(recv(hSocket,&serverResponse,1\*sizeof(char),0)<0)

{

printf("\nClient : - Receive failed");

return 1;

}

printf("\nClient : - Grade of student as sent by server is : %c",serverResponse);

close(hSocket);

return 0;

}

**Server :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nServer : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket)

{

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main()

{

int socket\_desc,client\_sock,c,read\_size,numberFromClient=0;

struct sockaddr\_in server, client;

char grade;

socket\_desc=createSocket();

if(socket\_desc == -1)

{

printf("\nServer : - Could not create socket");

return 1;

}

printf("\nServer : - Socket created");

if(bindCreatedSocket(socket\_desc) <0 )

{

printf("\nServer : - binding failed");

}

printf("\nServer : - binding done");

listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc, (struct sockaddr \*)&client, (socklen\_t\*)&c);

if (client\_sock < 0)

{

printf("\nServer : - Connection from client rejected.");

return 1;

}

printf("\nServer : - Connnection from client accepted");

if(recv(client\_sock, &numberFromClient, 1\*sizeof(int), 0) < 0 )

{

printf("\nServer : - Data uplink from client failed");

return 0;

}

printf("\nServer : - Marks send by client is :%d",numberFromClient);

if(numberFromClient >=85 && numberFromClient <= 100)

grade = 'A';

else if(numberFromClient >= 70 && numberFromClient < 85)

grade = 'B';

else if(numberFromClient >=60 && numberFromClient < 70)

grade = 'C';

else if(numberFromClient >=50 && numberFromClient <60)

grade = 'D';

else grade = 'F';

if(send(client\_sock, &grade,1\*sizeof(char),0)<0)

{

printf("\nServer : - Send Failed.");

return 1;

}

close(client\_sock);

return 0;

}

**Test Run:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc GradeC.c -o c

nitish89@ADMINRG-R5IIR8M ~

$ gcc GradeS.c -o s

nitish89@ADMINRG-R5IIR8M ~

$ ./s

Server : - Creating Socket

Server : - Socket created

Server : - binding done

Server : - Waiting for connections...

Server : - Connnection from client accepted

Server : - Marks send by client is :75

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./c

Client : - Creating Socket

Client : - Socket created

Client : - Successfully connected to socket

Client : - Enter marks obtained by a student : 75

Client : - Grade of student as sent by server is : B

**Q4. WAP using client-server socket programming: Client will enter two numbers n and m. The server will return the results of addition (a = n + m), substraction (b = n - m), multiplication (c = n** X **m) and division (d = n/m).**

**E.g., if n = 10 and m = 5**

**Output: a = n + m = 15, b = n – m = 5, c = n** X **m = 50, d = n/m = 2**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket; printf("\nClient : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectSocket(int hSocket){

int iRetval = -1, serverPort = 8080; struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr=inet\_addr("127.0.0.1"); remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket,(struct sockaddr \*)&remote,sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket, int \*number){

int shortRetVal = -1; shortRetVal = send(hSocket, number, 2\*sizeof(int), 0);

return shortRetVal;

}

int main(){

int hSocket,read\_size,numberToServer[2],serverResponse[4];

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket==-1) {printf("\nClient : - Could not create socket.");return 1; }

printf("\nClient : - Socket created.");

if(connectSocket(hSocket)<0){printf("\nClient : - Connection to socket failed.");return 1;}

printf("\nClient : - Successfully connected to socket");

printf("\nClient : - Enter the 1st number "); scanf("%d",&numberToServer[0]);

printf("\nClient : - Enter the 2nd number "); scanf("%d",&numberToServer[1]);

if(sendDataUsingSocket(hSocket,numberToServer)<0){ printf("Send failed\n"); return 1;}

if(recv(hSocket,&serverResponse,(4)\*sizeof(int),0)<0){printf("\nClient : - Data downlink failed.");return 1;}

printf("\nServer : - "); printf("\nAddition = %d ",serverResponse[0]);

printf("\nSubtraction = %d ",serverResponse[1]);printf("\nMultiplication = %d ",serverResponse[2]);

printf("\nDivision = %d ",serverResponse[3]);

close(hSocket);

return 0;

}

**Server :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<stdlib.h>

#include<limits.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket; printf("Creating Socket\n");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET; remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

void calculator(int m, int n,int ans[]){

ans[0] = m + n; ans[1] = m - n; ans[2] = m \* n;

if( n == 0 ) ans[3] = INT\_MAX;

else ans[3] = m/n;

}

int main(){

int socket\_desc,client\_sock,c,read\_size,numberFromClient[2]={0},answer[100];

struct sockaddr\_in client; socket\_desc = createSocket();

if(socket\_desc == -1) {

printf("\nServer : - Could not create socket.");return 1;}

printf("\nServer : - Socket created.");

if(bindCreatedSocket(socket\_desc) <0 ){printf("\nServer : - binding failed"); return 1;}

printf("\nServer : - binding done. "); listen(socket\_desc, 3);printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);client\_sock = accept(socket\_desc, (struct sockaddr \*)&client, (socklen\_t\*)&c);

if (client\_sock < 0){

printf("\nServer : - Incoming connection from client rejected."); return 1; }

printf("\nServer : - Incoming connection from client accepted.");

if(recv(client\_sock, &numberFromClient, 2\*sizeof(int), 0) < 0 ){

printf("\nServer : - Reception of data sent by client Failed.");return 1;}

calculator(numberFromClient[0],numberFromClient[1],answer);

if(send(client\_sock, answer, sizeof(answer)/sizeof(int), 0 ) < 0 ){

printf("\nServer : - Data downlink to client failed.");return 1;}

close(client\_sock);

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc CalC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc CalS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket

Server : - Socket created.

Server : - Waiting for connections...

Server : - Incoming connection from client accepted.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket.

Client : - Successfully connected to socket

Client : - Enter the 1st number 20

Client : - Enter the 2nd number 10

Server : - Addition = 30 Subtraction = 10

Multiplication = 200 Division = 2

**Q5. WAP using client-server socket programming: Client will enter an integer number n and server will return the result with sequence of \*.**

**E.g., if n = 3**

**Output:**

**\*\*\***

**\*\***

**\***

**Client:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket)

{

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket, int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main()

{

int hSocket, read\_size,numberToServer = 0;

struct sockaddr\_in server;

char serverResponse[100];

hSocket = createSocket();

if(hSocket == -1){

printf("\nClient : - Socket creation failed.");

return 1;

}

printf("\nClient : - Socket created successfully.");

if(connectToSocket(hSocket)<0){

printf("\nClient : - Socket connection Failed.");

return 1;

}

printf("\nClient : - Socket connected successfully. \nClient : - Enter a number :");

scanf("%d", &numberToServer);

if(numberToServer<0){

printf("\nClient : - Invalid input. Exiting.");

return 1;

}

if(sendDataUsingSocket(hSocket,&numberToServer)<0)

{

printf("\nClient : - Data uplink failed.");

return 1;

}

if(recv(hSocket,&serverResponse,(100)\*sizeof(char),0)<0)

{

printf("\nClient : - Data downlink failed.");

return 1;

}

printf("\nServer : - \n");

int j = 0,k = 1,t = numberToServer;

while(numberToServer != 0 )

{

for(int i=0; i < numberToServer; ++i)

printf("%c ",serverResponse[j++]);

printf("\n");

for(int j=0; j < k; ++j) printf(" ");

k++;numberToServer--;

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket)

{

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main()

{

int socket\_desc, client\_sock, c, read\_size,numberFromClient = 0;

struct sockaddr\_in server, client;

char serverResponse[100];

socket\_desc = createSocket();

if(socket\_desc == -1)

{

printf("\nServer : - Socket creation failed.");

return 1;

}

printf("\nServer : - Socket created.");

if(bindCreatedSocket(socket\_desc) <0 )

{

printf("\nServer : - Socket binding failed.");

return 1;

}

printf("\nServer : - Socket binding successful.");

listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if (client\_sock<0)

{

printf("\nServer : - Client connection request rejected.");

return 1;

}

printf("\nServer : - Client connection request accepted.");

if(recv(client\_sock,&numberFromClient,1\*sizeof(int),0)<0)

{

printf("\nServer : - Data uplink failed.");

return 0;

}

printf("\nServer : - The number send by client is :%d",numberFromClient);

int sum = (numberFromClient \* (numberFromClient + 1 )) /2;

for(int i=0; i < sum; ++i) serverResponse[i] = '\*';

if(send(client\_sock, serverResponse, sizeof(serverResponse)/sizeof(char),0)<0)

{

printf("\nServer : - Data donwlink failed.");

return 1;

}

close(client\_sock);

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc StarC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc StarS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - Socket created.

Server : - Socket binding successful.

Server : - Waiting for connections...

Server : - Client connection request accepted.

Server : - The number send by client is :3

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created successfully.

Client : - Socket connected successfully.

Client : - Enter a number : 3

Server : -

\* \* \*

\* \*

\*

**Q6. WAP using client-server socket programming: A client enters 0,1,2,3,4 in sequence and server returns the results 3,3,7,15,27 respectively in sequence (One by one)**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket)

{

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket,int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main()

{

int hSocket,read\_size,numberToServer,serverResponse,n=0,i=1;

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket == -1)

{

printf("\nClient : - Could not create socket.");

return 1;

}

printf("\nClient : - Socket created.");

if(connectToSocket(hSocket) < 0 )

{

printf("\nClient : - Connection to socket failed.");

return 1;

}

printf("\nClient : - Connection to socket successful.");

printf("\nClient : - Enter numbers in sequence : ");

while(i<=5)

{

if(i==1) printf("\nClient : - Enter the %d-st number : ",i);

else if(i==2) printf("\nClient : - Enter the %d-nd number : ",i);

else if(i==3) printf("\nClient : - Enter the %d-rd number : ",i);

else printf("\nClient : - Enter the %d-th number : ",i);

scanf("%d",&n); numberToServer = n;

if( sendDataUsingSocket(hSocket,&numberToServer)<0)

{

printf("\nClient : - Data uplink failed.");

return 1;

}

if(recv(hSocket,&serverResponse,1\*sizeof(int),0)<0)

{

printf("\nClient : - Data downlink failed.");

return 1;

}

printf("\nServer : - %d",serverResponse);

i++;

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket()

{

short hSocket;

printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket)

{

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main()

{

int socket\_desc, client\_sock, c, read\_size,numberFromClient = 0,i=1;

struct sockaddr\_in server, client;

int serverResponse[5] = {3,3,7,15,27};

socket\_desc = createSocket();

if(socket\_desc == -1)

{

printf("\nServer : - Socket creation failed.");

return 1;

}

printf("\nServer : - Socket created successfully.");

if(bindCreatedSocket(socket\_desc)<0)

{

printf("\nServer : - binding failed.");

return 1;

}

printf("\nServer : - binding successful.");

listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock<0)

{

printf("\nServer : - Connection request from client rejected.");

return 1;

}

printf("\nServer : - Connection request from client accepted\n");

while(i<=5)

{

if(recv(client\_sock,&numberFromClient,1\*sizeof(int),0)<0)

{

printf("\nServer : - Data uplink failed.");

return 1;

}

printf("\nServer : - The number send by client is :%d",numberFromClient);

if(send(client\_sock,&serverResponse[numberFromClient],1\*sizeof(int),0)<0)

{

printf("\nServer : - Data downlink failed.");

return 1;

}

i++;

}

close(client\_sock);

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc NumSeqOneByOneC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc NumSeqOneByOneS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - Socket created successfully.

Server : - binding successful.

Server : - Waiting for connections...

Server : - Connection request from client accepted

Server : - The number send by client is :0

Server : - The number send by client is :1

Server : - The number send by client is :2

Server : - The number send by client is :3

Server : - The number send by client is :4

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created.

Client : - Connection to socket successful.

Client : - Enter numbers in sequence :

Client : - Enter the 1-st number : 0

Server : - 3

Client : - Enter the 2-nd number : 1

Server : - 3

Client : - Enter the 3-rd number : 2

Server : - 7

Client : - Enter the 4-th number : 3

Server : - 15

Client : - Enter the 5-th number : 4

Server : - 27

**Q7. WAP using client-server socket programming: A client sends a composite number to the server and the server replies with the prime factorization of the number. Connection should not terminate till the client wants to.**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket){

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket,int \*number){

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main(){

int hSocket,numberToServer,i,serverResponse[100],factors;

hSocket = createSocket();

if(hSocket == -1){

printf("\nClient : - Could not create socket.");return 1;}

printf("\nClient : - Socket created.");

if(connectToSocket(hSocket) < 0 ){

printf("\nClient : - Connection to socket failed.");return 1;}

printf("\nClient : - Connection to socket successful.");

for(i=0;i<100;i++) serverResponse[i] = 0;

while(1){

printf("\nClient : - Enter a composite number(enter '0' to teminate connection) : ");

scanf("%d",&numberToServer);

if(numberToServer==0) {

printf("\nterminating connection on client request...");

if(sendDataUsingSocket(hSocket,&numberToServer)<0) {

printf("\nClient : - Data uplink failed."); return 1;}

return 0;

}

if(sendDataUsingSocket(hSocket,&numberToServer)<0){

printf("\nClient : - Data uplink failed."); return 1;}

if(recv(hSocket,&factors,1\*sizeof(int),0)<0){

printf("\nClient : - Data downlink failed.");return 1;}

if(recv(hSocket,serverResponse,100,0)<0){

printf("\nClient : - Data downlink failed.");return 1;}

printf("\nServer : - Prime factors for the numbers : ");

for(i=0;i<factors;i++) printf("%d ",serverResponse[i]);

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<math.h>

int factors,pf[100];

short createSocket(){

short hSocket;

printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET; remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

void primeFactors(int n) {

int i,j,k=0,isPrime;

for(i=2;i<=n;i++){

if(n%i==0){

isPrime = 1;

for(j=2;j<=i/2;j++){

if(i%j==0){ isPrime = 0;break;}

}

if(isPrime==1){

pf[k]=i; k++;

}

}

}

factors=k;

}

int main(){

int socket\_desc, client\_sock, c, numberFromClient,serverResponse[100],i;

struct sockaddr\_in client;

socket\_desc = createSocket();

if(socket\_desc == -1){

printf("\nServer : - Socket creation failed.");

return 1;

}

printf("\nServer : - Socket created successfully.");

if(bindCreatedSocket(socket\_desc)<0){

printf("\nServer : - binding failed.");

return 1;

}

printf("\nServer : - binding successful.");

listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock<0){

printf("\nServer : - Connection request from client rejected.");

return 1;

}

printf("\nServer : - Connection request from client accepted\n");

while(1){

if(recv(client\_sock, &numberFromClient, 1\*sizeof(int), 0) < 0 ){

printf("\nServer : - Data uplink from client failed.");

return 0;

}

printf("\nServer : - The number send by client is :%d",numberFromClient);

if(numberFromClient==0){

printf("\nServer : - Connection termination requested. Closing Session.");

close(client\_sock);return 0;

}

primeFactors(numberFromClient);

for(i=0;i<factors;i++) serverResponse[i] = pf[i];

printf("\nServer : - Prime factors for the numbers : ");

for(i=0;i<factors;i++)

printf("%d ",serverResponse[i]);

if(send(client\_sock,&factors,1\*sizeof(int),0)<0) {

printf("\nServer : - Data downlink failed.");

return 1;

}

if(send(client\_sock,serverResponse,100,0)<0){

printf("\nServer : - Data downlink failed.");

return 1;

}

}

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc PrimeFactS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - Socket created successfully.

Server : - binding successful.

Server : - Waiting for connections...

Server : - Connection request from client accepted

Server : - The number send by client is :99

Server : - The number send by client is :101

Server : - The number send by client is :0

Server : - Connection termination requested. Closing Session.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc PrimeFactC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created.

Client : - Connection to socket successful.

Client : - Enter a composite number(enter '0' to teminate connection) : 99

Server : - Prime factors for the numbers : 3 11

Client : - Enter a composite number(enter '0' to teminate connection) : 101

Server : - Prime factors for the numbers : 101

Client : - Enter a composite number(enter '0' to teminate connection) : 0

terminating connection on client request...

**Q8. WAP using client-server socket programming: Server maintains the database of students (at least 10 students) with the roll number as key. Client sends a roll number to server and server replies with all the corresponding information (at least 5 details). Connection should not terminate till the client wants to.**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket){

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket,int \*number){

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main(){

int hSocket,read\_size,numberToServer,n=0,i=1;

char serverResponse[100]={};

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket == -1){

printf("\nClient : - Could not create socket.");

return 1;

}

printf("\nClient : - Socket created.");

if(connectToSocket(hSocket) < 0 ){

printf("\nClient : - Connection to socket failed.");

return 1;

}

printf("\nClient : - Connection to socket successful.");

printf("\n\*\*\*\*\*\*Query Students Database with 10 students record\*\*\*\*\*\*");

printf("\n\*\*\*\*\*\*w.r.t. roll numbers, unique for each student(0-9)\*\*\*\*\*\*");

printf("\nN:B: Enter '99' against roll number to terminate querying");

while(1){

printf("\nClient : - Enter any roll number : ");

scanf("%d",&numberToServer);

if(numberToServer==99){

printf("\nterminating querying student database...");

if(sendDataUsingSocket(hSocket,&numberToServer)<0) {

printf("\nClient : - Data uplink failed.");

return 1;

}

return 0;

}

if(sendDataUsingSocket(hSocket,&numberToServer)<0) {

printf("\nClient : - Data uplink failed.");

return 1;

}

if(recv(hSocket,serverResponse,200,0)<0){

printf("\nClient : - Data downlink failed."); return 1;

}

printf("\nServer : - Corresponding student details :\n");

printf("%s",serverResponse);

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket;

printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main(){

int socket\_desc, client\_sock, c, read\_size,numberFromClient = -1,i=1;

struct sockaddr\_in server, client;

char serverResponse[100]={};

char\* sdb[10];

sdb[0] = "Name=Nitish\_Bhattacharjee\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Kolkata";

sdb[1] = "Name=Debjit\_Das\_\_\_Stream=IT\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Malda";

sdb[2] = "Name=Smaranjit\_Maiti\_\_\_Stream=ECE\_\_\_Year=3rd\_\_\_Semester=2nd\_\_\_Centre=Midnapore";

sdb[3] = "Name=K\_Charan\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Vizag";

sdb[4] = "Name=Sourav\_Bhadra\_\_\_Stream=IT\_\_\_Year=4th\_\_\_Semester=2nd\_\_\_Centre=Beleghata";

sdb[5] = "Name=Ashish\_Sarkar\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Agartala" ;

sdb[6] = "Name=Souvik\_Sen\_\_\_Stream=ECE\_\_\_Year=3rd\_\_\_Semester=1st\_\_\_Centre=Kolkata";

sdb[7] = "Name=Manoj\_Bose\_\_\_Stream=IT\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Jalpaiguri";

sdb[8] = Name=Arindam\_Raymondal\_\_\_Stream=ECE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Alipurduar";

sdb[9] = "Name=Asit\_Mahato\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Bankura";

socket\_desc = createSocket();

if(socket\_desc == -1){printf("\nServer : - Socket creation failed.");return 1;}

printf("\nServer : - Socket created successfully.");

if(bindCreatedSocket(socket\_desc)<0){ printf("\nServer : - binding failed."); return 1;}

printf("\nServer : - binding successful."); listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections..."); c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock<0){

printf("\nServer : - Connection request from client rejected."); return 1;

}

printf("\nServer : - Connection request from client accepted\n");

while(1){

if(recv(client\_sock,&numberFromClient,1\*sizeof(int),0)<0){

printf("\nServer : - Data uplink failed."); return 1;

}

printf("\nServer : - The number send by client is :%d",numberFromClient);

if(numberFromClient==99){

printf("\nServer : - Connection termination requested. Closing Session.");

close(client\_sock); return 0;

}

strcpy(serverResponse,sdb[numberFromClient]);

if(send(client\_sock,serverResponse,strlen(serverResponse),0)<0) {

printf("\nServer : - Data downlink failed."); return 1;

}

}

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc StudentdbS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ gcc StudentdbC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - binding successful.

Server : - Waiting for connections...

Server : - The number send by client is :3

server will return : Name=K\_Charan\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Vizag

Server : - The number send by client is :1

server will return : Name=Debjit\_Das\_\_\_Stream=IT\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Malda

Server : - The number send by client is :99

Server : - Connection termination requested. Closing Session.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created.

Client : - Connection to socket successful.

\*\*\*\*\*\*Query Students Database with 10 students record\*\*\*\*\*\*

\*\*\*\*\*\*w.r.t. roll numbers, unique for each student(0-9)\*\*\*\*\*\*

N:B: Enter '99' against roll number to terminate querying

Client : - Enter any roll number : 3

Server : - Corresponding student details :

Name=K\_Charan\_\_\_Stream=CSE\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Vizag

Client : - Enter any roll number : 1

Server : - Corresponding student details :

Name=Debjit\_Das\_\_\_Stream=IT\_\_\_Year=2nd\_\_\_Semester=2nd\_\_\_Centre=Malda

Client : - Enter any roll number : 99

terminating querying student database...

**Q9. WAP using client-server socket programming: Client sends a word and a number to server and server sends the letter which occurs the given number of times in the given word, if it exists else sends the letter which occurs maximum number of times in the given word. Connection should not terminate till the client wants to.**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket){

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendUsingSocket(int hSocket, int \*number){

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int sendWordDataUsingSocket(int hSocket,char \*s, int len){

int shortRetVal = -1;

shortRetVal = send(hSocket,s,len,0);

return shortRetVal;

}

int main(){

int hSocket,numberToServer,numberFromServer=0;

char letterFromServer,wordToServer[25];

struct sockaddr\_in server;

hSocket = createSocket();

if(hSocket == -1){

printf("\nClient : - Could not create socket."); return 1; }

printf("\nClient : - Socket created.");

if(connectToSocket(hSocket) < 0 ){

printf("\nClient : - Connection to socket failed.");return 1; }

printf("\nClient : - Connection to socket successful.");

while(1){

printf("\nClient : - Enter the word(enter 'X'/'x' to terminate) : ");scanf("%s",wordToServer);

if(!(strcmp(wordToServer,"X"))||!(strcmp(wordToServer,"x"))) {

printf("\nClient : - Session termination has been requested. Closing session.");

if(sendWordDataUsingSocket(hSocket,wordToServer,25)<0){

printf("\nClient : - Data uplink failed."); return 1; }

return 0; }

if(sendWordDataUsingSocket(hSocket,wordToServer,25)<0){

printf("\nClient : - Data uplink failed."); return 1; }

printf("\nClient : - Enter the occurence count : ");scanf("%d",&numberToServer);

if(sendUsingSocket(hSocket, &numberToServer)<0){

printf("\nClient :- Data uplink failed\n"); return 1; }

if(recv(hSocket,&letterFromServer,1\*sizeof(char),0)<0) {

printf("\nClient : - Data downlink failed."); return 1;}

if(recv(hSocket,&numberFromServer,1\*sizeof(int),0)<0){

printf("\nClient : - Data downlink failed."); return 1;}

{

if(numberFromServer==numberToServer)

printf("\nServer : - In the word %s the letter occuring %d times is %c",wordToServer,numberFromServer,letterFromServer);

else

{

printf("\nServer : - In the word %s the no letter occurs the suggested %d times..",wordToServer,numberToServer);

printf("\n\tHence showing the letter having maximum occurence(=%d) : %c",numberFromServer,letterFromServer);

}

}

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

#include <ctype.h>

int numberToClient;

short createSocket(){

short hSocket;

printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET; remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

char findOccurence(char \*word,int num){

char c;

int i,freq[26],occured,flag=0,m=0;

for(i=0;i<26;i++) freq[i]=0;

for(i=0;i<strlen(word);i++){

occured = (int)(toupper(word[i])-'A');freq[occured]++;

}

for(i=0;i<26;i++){

if(freq[i]==num){c = (char)(65+i);numberToClient=num; flag = 1;}

else if(freq[i]>freq[m]) m = i; else continue;

}

if(flag==0){c = (char)(65+m);numberToClient=freq[m]; }

return c;

}

int main(){

int socket\_desc,client\_sock,numberFromClient=0,c;

char letterToClient,wordFromClient[25];

struct sockaddr\_in client;

socket\_desc = createSocket();

if(socket\_desc == -1){printf("\nServer : - Socket creation failed."); return 1;}

printf("\nServer : - Socket created successfully.");

if(bindCreatedSocket(socket\_desc)<0){ printf("\nServer : - binding failed."); return 1;}

printf("\nServer : - binding successful.");

listen(socket\_desc, 3); printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock<0){

printf("\nServer : - Connection request from client rejected."); return 1; }

printf("\nServer : - Connection request from client accepted\n");

while(1){

if(recv(client\_sock,wordFromClient,25,0)<0){

printf("\nServer : - Data uplink failed."); return 1; }

printf("\nServer : - The word send by client is :%s",wordFromClient);

if(!(strcmp(wordFromClient,"X"))||!(strcmp(wordFromClient,"x"))){

printf("\nServer : - Connection termination requested. Closing Session.");

close(client\_sock); return 0;}

if(recv(client\_sock,&numberFromClient,1\*sizeof(int),0)<0){

printf("\nServer : - Data uplink failed."); return 1; }

letterToClient = findOccurence(wordFromClient,numberFromClient);

if(send(client\_sock,&letterToClient,1\*sizeof(char),0)<0) {

printf("\nServer : - Data downlink failed.");return 1; }

if(send(client\_sock,&numberToClient,1\*sizeof(char),0)<0){

printf("\nServer : - Data downlink failed.");return 1; }

}

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc WordLetterOccurC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created.

Client : - Connection to socket successful.

Client : - Enter the word(enter 'X'/'x' to terminate) : Test

Client : - Enter the occurence count : 2

Server : - In the word Test the letter occuring 2 times is : T

Client : - Enter the word(enter 'X'/'x' to terminate) : Banana

Client : - Enter the occurence count : 4

Server : - In the word Banana the no letter occurs the suggested 4 times..

Hence showing the letter having maximum occurence(=3) : A

Client : - Enter the word(enter 'X'/'x' to terminate) : X

Client : - Session termination has been requested. Closing session.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc WordLetterOccurS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - Socket created successfully.

Server : - binding successful.

Server : - Waiting for connections...

Server : - Connection request from client accepted

Server : - The word send by client is :Test

Server : - The word send by client is :Banana

Server : - The word send by client is :X

Server : - Connection termination requested. Closing Session.

**Q10. WAP using client-server socket programming: Server has the details for each employee, about the number of leaves that an employee can take at present (total leaves; current balance of leaves; extra leaves). Client sends an employee ID to server with the number of leaves he/she wants to take and server replies with the appropriate response. Connection should not terminate till the client wants to.**

**Client :**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<strings.h>

#include<string.h>

#include<netdb.h>

#include<unistd.h>

void error(char \*msg)

{

perror(msg);

exit(0);

}

int main(int argc, char \*argv[])

{

int sockfd,portno,n,id;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char buffer[256];

if(argc < 3)

{

fprintf(stderr,"\nClient : Correct execution format is %s <hostname> <port\_no> {e.g. %s localhost 2222}", argv[0]);

exit(0);

}

portno = atoi(argv[2]);

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd < 0)

error("\nClient : Could not create socket.");

printf("\nClient : Socket created.");

server = gethostbyname(argv[1]);

if(server == NULL)

{

fprintf(stderr,"\nClient : Could n't find server as given");

exit(0);

}

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,(char \*)&serv\_addr.sin\_addr.s\_addr,server->h\_length);

serv\_addr.sin\_port = htons(portno);

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

error("\nClient : Connection to socket failed.");

printf("\nClient : Connection to socket successful.");

while(1)

{

printf("\nClient : Enter the Emp ID(enter -1 to terminate session) : ");

bzero(buffer,256);

fgets(buffer,255,stdin);

id = atoi(buffer);

if(id==-1) error("\nClient : Session termination has been requested");

n = write(sockfd,buffer,strlen(buffer));

if(n<0) error("\nClient : Couldn't write the data input to socket.");

bzero(buffer,256);

n = read(sockfd,buffer,255);

printf("\nClient : Requesting mumber of leaves ? : ");bzero(buffer,256);

fgets(buffer,255,stdin);

n = write(sockfd,buffer,strlen(buffer));

bzero(buffer,256);

n = read(sockfd,buffer,255);

if (n < 0) error("\nClient : Data downlink from server failed.");

printf("\nServer : %s\n",buffer);

}

return 0;

}

**Server:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<strings.h>

#include<string.h>

#include<unistd.h>

void error(char \*msg)

{

perror(msg);

exit(1);

}

int main(int argc, char \*argv[])

{

int sockfd, newsockfd, portno, clilen,i,id,number,total = 10,remain[100],extra[100],n;

char buffer[256];

struct sockaddr\_in serv\_addr, cli\_addr;

if(argc<2)

{

fprintf(stderr,"\nServer : Provide port no. Usage: %s <port\_no> {e.g. %s 2222}",argv[0],argv[0]);

exit(1);

}

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) error("\nServer : Couldn't create socket.");

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

portno = atoi(argv[1]);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(portno);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0) error("\nSocket binding failed.");

listen(sockfd,5);

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd, (struct sockaddr \*) &cli\_addr, &clilen);

if(newsockfd<0) error("\nServer : Incoming client connection rejected.");

for(i=0;i<100;i++)

{

remain[i] = i%10;extra[i] = 0;

}

while(1)

{

bzero(buffer,256);

n = read(newsockfd,buffer,255);

if (n < 0) error("\nServer : Couldn't read from socket ");

id = atoi(buffer);

if(id==-1) error("\nServer : ");

id = id%100;

bzero(buffer,256);

strcpy(buffer,"EmpID Recieved");

n = write(newsockfd,buffer,18);

if(n<0) error("\nServer : Couldn't write to socket.");

bzero(buffer,256);

n = read(newsockfd,buffer,255);

if (n < 0) error("\nServer : Couldn't read from socket ");

number = atoi(buffer);

bzero(buffer,256);

if(number>remain[id])

{

extra[id] = extra[id] + (number - remain[id]);

sprintf(buffer,"%d",(number - remain[id]));

strcat(buffer," of the leaves requested are not available. Total Leave for all employees = 10. Confirming all available leaves. Current balance leaves = 0");

}

else

{

remain[id]-=number;

sprintf(buffer,"%d",(remain[id]));

strcat(buffer," leaves are still remaining. Total leave for all employees = 10. All applied leaves are confirmed.");

}

n = write(newsockfd,buffer,strlen(buffer));

if (n < 0) error("\nServer : Couldn't write to the socket");

}

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc EmployeeLeaveS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S 5878

Server : Couldn't read from socket : Software caused connection abort

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc EmployeeLeaveC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ ./C localhost 5878

Client : Socket created.

Client : Connection to socket successful.

Client : Enter the Emp ID(enter -1 to terminate session) : 857

Client : Requesting mumber of leaves ? : 11

Server : 4 of the leaves requested are not available. Total Leave for all employees = 10. Confirming all available leaves. Current balance leaves = 0

Client : Enter the Emp ID(enter -1 to terminate session) : 78

Client : Requesting mumber of leaves ? : 4

Server : 4 leaves are still remaining. Total leave for all employees = 10. All applied leaves are confirmed.

Client : Enter the Emp ID(enter -1 to terminate session) : -1

Client : Session termination has been requested: No error

**Q11. WAP using client-server socket programming: Client has a list of words (at least 20) and server has a dictionary (at least 15 entries). Client picks up a random word from its list and sends it to server. Server responds with its meaning, if found else adds this entry to its dictionary with its meaning.**

**Client :**

#include<sys/socket.h>

#include<stdio.h>

#include<time.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<stdlib.h>

#include<ctype.h>

#include<time.h>

#include<string.h>

void error(char \*msg)

{

perror(msg);

exit(0);

}

short createSocket()

{

short hSocket;

printf("\nClient : Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectSocket(int hSocket)

{

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendUsingSocket(int hSocket, int \*number)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, number,1\*sizeof(int), 0);

return shortRetVal;

}

int sendDataUsingSocket(int hSocket, char \*message, int len)

{

int shortRetVal = -1;

shortRetVal = send(hSocket, message, len, 0);

return shortRetVal;

}

int main()

{

int hSocket,n;

char \*WordList[20] = {"aglow","balderdash","castellan", "frowzy", "frugal",

"larceny","malady","smutty","torpid","torrent","transfix","uncalled","waterproof","worm-

eaten","yam","yawn","yell","yuletide","zephyr","zealot"};

char \*meaning,means[100];

hSocket = createSocket();

if(hSocket == -1) error("\nClient : Could not create socket.");

printf("\nClient : Socket created successfully.");

if(connectSocket(hSocket)<0) error("\nClient : Could not connect to socket.");

printf("\nClient : Successfully connected with socket");

printf("\n=====\*\*\*\*\*20 Words in Client, 15 words in Server Dictionary\*\*\*\*\*=====");

printf("\n 1.aglow\t 2.balderdash\t 3.castellan\t 4.frowzy\t 5.frugal\n 6.larceny\t 7.malady\t 8.smutty\t 9.torpid\t10.torrent");

printf("\n11.transfix\t12.uncalled\t13.waterproof\t14.worm-eaten\t15.yam\n16.yawn\t\t17.yell\t\t18.yuletide\t19.zephyr\t20.zealot");

srand(time(0));n=rand()%20;

printf("\nClient : Random word selected : %s",WordList[n]);

if(sendUsingSocket(hSocket,&n)<0) error("\nClient : Could not send data using socket.");

if(recv(hSocket,meaning,500,0)<0) error("\nClient : Could not receive data using socket.");

printf("\nServer : %s",meaning);

if(n>14)

{

printf("\nClient : Enter meaning of the word :");scanf("%s",means);

if(sendDataUsingSocket(hSocket,means,100)<0)

{

printf("\nClient : - Data uplink from client failed. Exiting.");

return 1;

}

printf("\nClient : New meaning is send to be updated in the dictionary.");

}

close(hSocket);

return 0;

}

**Server:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<stdlib.h>

void error(char \*msg){

perror(msg);

exit(0);

}

short createSocket(){

short hSocket;

printf("\nServer : Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET;

remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main(){

char \*MeaningList[20] = {"red-hot","nonsense","governor\_of\_a\_castle","ill- smelling"," economical",

"theft","mental\_or\_bodily\_ illness","obscene","inactive","a\_rush\_of\_water","pierce\_through","uninvited",

"impervious\_to\_water","gnawed\_by\_insects","a\_kind\_of\_sweet\_potato","Meaning\_not\_available",

"Meaning\_not\_available","Meaning\_not\_available","Meaning\_not\_available","Meaning\_not\_available"};

char \*message={0};

int socket\_desc,client\_sock,c,numberFromClient;

struct sockaddr\_in server, client;

socket\_desc=createSocket();

if(socket\_desc == -1) error("\nServer : Could not create socket.");

printf("\nServer : Socket created.");

if(bindCreatedSocket(socket\_desc)<0) error("\nServer : Binding failed.");

printf("\nServer : Socket binding done.");

listen(socket\_desc, 3);

printf("\nServer : Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc, (struct sockaddr \*)&client, (socklen\_t\*)&c);

if(client\_sock<0) error("\nServer : Client connection rejected");

printf("\nServer : Client connnection accepted.");

if(recv(client\_sock, &numberFromClient, 1\*sizeof(int),0)<0) error("\nServer : Data reception from client failed.");

if(send(client\_sock,MeaningList[numberFromClient],200,0)<0) error("\nServer: Send data to Client failed");

if(numberFromClient>14)

printf("\nServer : Meaning not in dictionary.");

close(client\_sock);

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc DictionaryS.c -o ds

nitish89@ADMINRG-R5IIR8M ~

$ ./ds

Server : Creating Socket

Server : Socket created.

Server : Waiting for connections...

Server : Client connnection accepted.

Server : Meaning not in dictionary.

nitish89@ADMINRG-R5IIR8M ~

$ ./ds

Server : Creating Socket

Server : Socket created.

Server : Waiting for connections...

Server : Client connnection accepted.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc DictionaryC.c -o dc

nitish89@ADMINRG-R5IIR8M ~

$ ./dc

Client : Creating Socket.

Client : Socket created successfully.

Client : Successfully connected with socket

=====\*\*\*\*\*20 Words in Client, 15 words in Server Dictionary\*\*\*\*\*=====

1.aglow 2.balderdash 3.castellan 4.frowzy 5.frugal

6.larceny 7.malady 8.smutty 9.torpid 10.torrent

11.transfix 12.uncalled 13.waterproof 14.worm-eaten 15.yam

16.yawn 17.yell 18.yuletide 19.zephyr 20.zealot

Client : Random word selected : zealot

Server : Meaning\_not\_available

Client : Enter meaning of the word :fanatic

Client : New meaning is send to be updated in the dictionary.

nitish89@ADMINRG-R5IIR8M ~

$ ./dc

Client : Creating Socket.

Client : Socket created successfully.

Client : Successfully connected with socket

=====\*\*\*\*\*20 Words in Client, 15 words in Server Dictionary\*\*\*\*\*=====

1.aglow 2.balderdash 3.castellan 4.frowzy 5.frugal

6.larceny 7.malady 8.smutty 9.torpid 10.torrent

11.transfix 12.uncalled 13.waterproof 14.worm-eaten 15.yam

16.yawn 17.yell 18.yuletide 19.zephyr 20.zealot

Client : Random word selected : aglow

Server : red-hot

**Q12. WAP using client-server socket programming: Server stores set of questions and their answers in its databsase(at least 10). Client sends a question to server and server replies with the corresponding answer, if found else reply as try again. Connection should terminate after client has send three questions or client wants.**

**Server :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket; printf("\nServer : - Creating Socket.");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET; remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main(){

int socket\_desc, client\_sock, c, numberFromClient,i=1;

struct sockaddr\_in client; char serverResponse[100]={};

char\* sdb[10];sdb[0] = "New\_Delhi";sdb[1] = "Twenty\_Nine";

sdb[2] = "Ram\_Nath\_Kovind"; sdb[3] = "Rajendra\_Prasad";sdb[4] = "Indian\_Peafowl";

sdb[5] = "Bengal\_Tiger" ;sdb[6] = "Rabindranath\_Tagore"; sdb[7] = "Mahatma\_Gandhi"; sdb[8] = "1947";sdb[9] = "Lotus";

socket\_desc = createSocket();

if(socket\_desc == -1){

printf("\nServer : - Socket creation failed.");

return 1;

}

printf("\nServer : - Socket created successfully.");

if(bindCreatedSocket(socket\_desc)<0){

printf("\nServer : - binding failed.");

return 1;

}

printf("\nServer : - binding successful."); listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections..."); c = sizeof(struct sockaddr\_in);

client\_sock = accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock<0){

printf("\nServer : - Connection request from client rejected.");

return 1;

}

printf("\nServer : - Connection request from client accepted\n");

while(1){

if(recv(client\_sock,&numberFromClient,1\*sizeof(int),0)<0)

{

printf("\nServer : - Data uplink failed.");

return 1;}

printf("\nServer : - The number send by client is :%d",numberFromClient);

if(numberFromClient==99){

printf("\nServer : - Connection termination requested. Closing Session.");

close(client\_sock);

return 0;}

if((numberFromClient<0)||(numberFromClient>9)) strcpy(serverResponse,"Try\_Again...................");

else strcpy(serverResponse,sdb[numberFromClient]);

if(send(client\_sock,serverResponse,strlen(serverResponse),0)<0) {

printf("\nServer : - Data downlink failed.");

return 1;}

}

}

**Client:**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket;

printf("\nClient : - Creating Socket");

hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectToSocket(int hSocket){

int iRetval = -1,serverPort = 8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET;

remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket,int \*number){

int shortRetVal = -1;

shortRetVal = send(hSocket, number, 1\*sizeof(int), 0);

return shortRetVal;

}

int main(){

int hSocket,numberToServer,qn=0,n;

char serverResponse[100]={};

hSocket = createSocket();

if(hSocket == -1){

printf("\nClient : - Could not create socket.");

return 1;

}

printf("\nClient : - Socket created.");

if(connectToSocket(hSocket) < 0 ){

printf("\nClient : - Connection to socket failed.");

return 1;

}

printf("\nClient : - Connection to socket successful.");

printf("\n\*\*\*\*\*\*\*Question Bank\*\*\*\*\*\*\*\n\*\*\*\*enter 1 to 10 to choose from 10 questions\*\*\*\*");

printf("\n[1] What is the capital of India?\t[2] How many states are there in India?\n[3] What is the

name of president of India?");

printf("\t[4] Who was the first president of India?\n[5] What is the national bird of India?\t[6] What

is the national animal of India?");

printf("\n[7] Who wrote Indian national anthem?\t[8] Who is called the father of the nation?\n[9] On

which year did India get it's independence?");

printf("\t[10] What is the national flower of India?");

printf("\nN:B: Enter '99' to terminate querying Question Bank");

while(1){

qn++; printf("\nClient : - Question number ? : ");scanf("%d",&n); numberToServer=n-1;

if(n==99){

printf("\nterminating session on client request...");

if(sendDataUsingSocket(hSocket,&n)<0){

printf("\nClient : - Data uplink failed.");

return 1;

}

return 0;

}

if(qn>3){

n=99;

printf("\nterminating session , max question limit per-session(=3) exceeded.");

if(sendDataUsingSocket(hSocket,&n)<0){

printf("\nClient : - Data uplink failed."); return 1;

}

return 0;

}

if(sendDataUsingSocket(hSocket,&numberToServer)<0) {

printf("\nClient : - Data uplink failed."); return 1;}

if(recv(hSocket,serverResponse,200,0)<0){

printf("\nClient : - Data downlink failed.");return 1;}

printf("\nServer : - %s",serverResponse);

}

close(hSocket);

return 0;

}

**Test Run :**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc QuestionBankC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc QuestionBankS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket.

Server : - Socket created successfully.

Server : - binding successful.

Server : - Waiting for connections...

Server : - Connection request from client accepted

Server : - The number send by client is :0

Server : - The number send by client is :4

Server : - The number send by client is :124

Server : - The number send by client is :99

Server : - Connection termination requested. Closing Session.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created.

Client : - Connection to socket successful.

\*\*\*\*\*\*\*\*\*\*Question Bank\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*enter 1 to 10 to choose from 10 questions\*\*\*\*\*\*

[1] What is the capital of India? [2] How many states are there in India?

[3] What is the name of president of India? [4] Who was the first president of India?

[5] What is the national bird of India? [6] What is the national animal of India?

[7] Who wrote Indian national anthem? [8] Who is called the father of the nation?

[9] On which year did India get it's independence? [10] What is the national flower of India?

N:B: Enter '99' to terminate querying Question Bank

Client : - Question number ? : 1 Server : - New\_Delhi

Client : - Question number ? : 5 Server : - Indian\_Peafowl

Client : - Question number ? : 125 Server : - Try\_Again...................

Client : - Question number ? : 8

terminating session , max question limit per-session(=3) exceeded.

**Q14. WAP using client-server socket programming to download: Client will specify commands like ./a.out<server\_ip> <port\_no><get><filename>(Use different directory for client and server)**

**Client :**

#include<stdlib.h>

#include<stdio.h>

#include<errno.h>

#include<string.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<sys/wait.h>

#include<sys/socket.h>

#include<signal.h>

#include<ctype.h>

#include<arpa/inet.h>

#include<netdb.h>

#include<unistd.h>

#define LENGTH 512

void error(const char \*msg){perror(msg);exit(1); }

int main(int argc, char \*argv[]){

int sockfd,PORT, nsockfd;

char revbuf[LENGTH],\*filename;

struct sockaddr\_in remote\_addr;

if(argc < 5){

fprintf(stderr,"\nUsage : %s <hostname> <port\_no> <get> <filename>", argv[0]);

exit(0);

}

PORT = atoi(argv[2]);filename = argv[4];

if(strcmp(argv[1],"localhost"))

fprintf(stderr, "ERROR: Failed to connect to the server try with localhost");

exit(1);

}

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1){

fprintf(stderr, "ERROR: Failed to obtain Socket Descriptor! (errno = %d)\n",errno);

exit(1);

}

remote\_addr.sin\_family = AF\_INET; remote\_addr.sin\_port = htons(PORT);

inet\_pton(AF\_INET, "127.0.0.1", &remote\_addr.sin\_addr);

bzero(&(remote\_addr.sin\_zero), 8);

if (connect(sockfd, (struct sockaddr \*)&remote\_addr, sizeof(struct sockaddr)) == -1){

fprintf(stderr, "ERROR: Failed to connect to the host! (errno = %d)\n",errno);

exit(1);

}

else printf("Client : Connected to server at port %d...ok!\n", PORT);

if(strcmp(filename,"output.txt") error("\nERROR : File is not available in server");

char \*fr\_name="/home/nitish89/FileC/output.txt";

printf("\nfile path %s",fr\_name);

FILE \*fr = fopen(fr\_name, "a");

if(fr == NULL) printf("Error : File %s Cannot be opened.\n", fr\_name);

else{

bzero(revbuf, LENGTH);

int fr\_block\_sz = 0;

while((fr\_block\_sz = recv(sockfd, revbuf, LENGTH, 0)) > 0) {

int write\_sz = fwrite(revbuf, sizeof(char), fr\_block\_sz, fr);

if(write\_sz < fr\_block\_sz){error("Error : File write failed.\n"); }

bzero(revbuf, LENGTH);

if (fr\_block\_sz == 0 || fr\_block\_sz != 512) { break;}

}

if(fr\_block\_sz < 0) {

if (errno == EAGAIN){ printf("recv() timed out.\n");}

else{fprintf(stderr, "recv() failed due to errno = %d\n", errno);}

}

printf("Client : File Download completed from server\n");

fclose(fr);

}

close (sockfd);

printf("Client : Connection lost.\n");

return (0);

}

**Server :**

#include<stdlib.h>

#include<stdio.h>

#include<errno.h>

#include<string.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<sys/wait.h>

#include<sys/socket.h>

#include<signal.h>

#include<ctype.h>

#include<arpa/inet.h>

#include<netdb.h>

#include<unistd.h>

#define BACKLOG 5

#define LENGTH 512

void error(const char \*msg){perror(msg);exit(1); }

int main(int argc, char \*argv[]){

int sockfd,nsockfd,num,sin\_size,PORT;

struct sockaddr\_in addr\_local,addr\_remote;

char revbuf[LENGTH];

if(argc<2){

fprintf(stderr,"\nUsage : %s <port\_no>", argv[0]);

exit(0);

}

PORT = atoi(argv[1]);

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1 ){

fprintf(stderr, "ERROR: Failed to obtain Socket Descriptor. (errno = %d)\n", errno);

exit(1);

}

else printf("Server : Obtaining socket descriptor successfully.\n");

addr\_local.sin\_family = AF\_INET; addr\_local.sin\_port = htons(PORT);

addr\_local.sin\_addr.s\_addr = INADDR\_ANY;

bzero(&(addr\_local.sin\_zero), 8);

if( bind(sockfd, (struct sockaddr\*)&addr\_local, sizeof(struct sockaddr)) == -1 ){

fprintf(stderr, "ERROR: Failed to bind Port. (errno = %d)\n", errno);

exit(1);

}

else printf("Server : Binded tcp port %d in addr 127.0.0.1 sucessfully.\n",PORT);

if(listen(sockfd,BACKLOG) == -1){

fprintf(stderr, "ERROR: Failed to listen Port. (errno = %d)\n", errno);

exit(1);

}

else printf ("Server : Listening the port %d successfully.\n", PORT);

int success = 0;

while(success == 0){

sin\_size = sizeof(struct sockaddr\_in);

if((nsockfd = accept(sockfd, (struct sockaddr \*)&addr\_remote, &sin\_size)) == -1) {

fprintf(stderr, "ERROR: Obtaining new Socket Despcritor. (errno = %d)\n", errno);

exit(1);

}

else

printf("Server : Server has got connected from %s.\n", inet\_ntoa(addr\_remote.sin\_addr));

char\* fs\_name = "/home/nitish89/FileS/output.txt";

char sdbuf[LENGTH];

printf("\nServer : Sending %s to the Client...", fs\_name);

FILE \*fs = fopen(fs\_name, "r");

if(fs == NULL)

{

fprintf(stderr, "ERROR: File %s not found on server. (errno = %d)\n", fs\_name, errno);

exit(1);

}

bzero(sdbuf, LENGTH); int fs\_block\_sz;

while((fs\_block\_sz = fread(sdbuf, sizeof(char), LENGTH, fs))>0) {

if(send(nsockfd, sdbuf, fs\_block\_sz, 0) < 0) {

fprintf(stderr, "ERROR: Failed to send file %s. (errno = %d)\n", fs\_name, errno);

exit(1);

}

bzero(sdbuf, LENGTH);

}

printf("\nServer : File sent to client successfully!\n"); success = 1;

close(nsockfd);

printf("Server : Connection with Client closed. Server will wait now...\n");

while(waitpid(-1, NULL, WNOHANG) > 0);

}

}

**Test Run[1]:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ cd FileC

nitish89@ADMINRG-R5IIR8M ~/FileC

$ gcc FileDownloadC.c -o fc

nitish89@ADMINRG-R5IIR8M ~/FileC

$ ./fc localhost 2020 get FileS/output.txt

Client : Connected to server at port 2020...ok!

file path /home/nitish89/FileC/output.txt

Client : File Download completed from server

Client : Connection lost.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ cd FileS

nitish89@ADMINRG-R5IIR8M ~/FileS

$ gcc FileDownloadS.c -o fs

nitish89@ADMINRG-R5IIR8M ~/FileS

$ ./fs 2020

Server : Obtaining socket descriptor successfully.

Server : Binded tcp port 2020 in addr 127.0.0.1 sucessfully.

Server : Listening the port 2020 successfully.

Server : Server has got connected from 127.0.0.1.

Server : Sending /home/nitish89/FileS/output.txt to the Client...

Server : File sent to client successfully!

Server : Connection with Client closed. Server will wait now...

**Q16. WAP using client-server socket programming: Client needs to authenticate itself by entering a server defined string as a password and then to say Hi to server. Server replies with Hello. Press Any key to exit**

**Client :**

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket; hSocket = socket(AF\_INET, SOCK\_STREAM, 0);

return hSocket;

}

int connectSocket(int hSocket){

int iRetval=-1,serverPort=8080;

struct sockaddr\_in remote = {0};

remote.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

remote.sin\_family = AF\_INET; remote.sin\_port = htons(serverPort);

iRetval = connect(hSocket, (struct sockaddr \*)&remote, sizeof(struct sockaddr\_in));

return iRetval;

}

int sendDataUsingSocket(int hSocket, char \*message, int len){

int shortRetVal = -1;shortRetVal = send(hSocket, message, len, 0);

return shortRetVal;

}

int main(){

int hSocket,read\_size;struct sockaddr\_in server;

char sendToServer[100]={0},serverReply[200]={0}, \*invalid="Wrong Password!!!";

hSocket=createSocket();

if(hSocket == -1){printf("\nClient : - Could not create socket. Exiting.");return 1;}

printf("\nClient : - Socket created successfully.");

if(connectSocket(hSocket) < 0 ) {printf("\nClient : - Establishing connection to socket failed.");return 1;}

printf("\nClient : - Enter password to authenticate : ");scanf("%s",sendToServer);

if(sendDataUsingSocket(hSocket,sendToServer,strlen(sendToServer))<0){

printf("\nClient : - Data uplink from client failed. Exiting.");return 1;}

if(recv(hSocket,serverReply,200,0)<0){

printf("\nClient : - Data downlink from server failed.");return 1; }

printf("\n\nSever : - %s", serverReply);

if(strcmp(serverReply,invalid)==0) return 1;

memset(sendToServer,'\0',sizeof(sendToServer));memset(serverReply,'\0',sizeof(serverReply));

printf("\nChat with server"); printf("\nClient : - Enter message : "); scanf("%s",sendToServer);

if(sendDataUsingSocket(hSocket,sendToServer,strlen(sendToServer))<0){

printf("\nClient : - Data uplink from client failed. Exiting.");return 1;}

if(recv(hSocket,serverReply,200,0)<0){

printf("\nClient : - Data downlink from server failed. Exiting."); return 1;}

printf("\nServer : - %s",serverReply);

close(hSocket);

return 0;

}

**Server** :

#include<string.h>

#include<sys/socket.h>

#include<stdio.h>

#include<arpa/inet.h>

#include<unistd.h>

short createSocket(){

short hSocket; hSocket=socket(AF\_INET,SOCK\_STREAM,0);

return hSocket;

}

int bindCreatedSocket(int hSocket){

int iRetval=-1,ClientPort = 8080;

struct sockaddr\_in remote={0};

remote.sin\_family = AF\_INET; remote.sin\_addr.s\_addr = htonl(INADDR\_ANY);

remote.sin\_port = htons(ClientPort);

iRetval = bind(hSocket,(struct sockaddr \*)&remote,sizeof(remote));

return iRetval;

}

int main(){

int socket\_desc, client\_sock, c, read\_size;

struct sockaddr\_in server, client;

char client\_message[200]={0}, message[100] = {0}, msg[100] = {0};

const char \*pMessage = "nitish";

socket\_desc=createSocket();

if(socket\_desc==-1){printf("\nServer : - Could not create socket. Exiting.");return 1;}

printf("\nServer : - Socket created.");

if(bindCreatedSocket(socket\_desc)<0){

printf("\nServer : - socket binding failed.");

}

printf("\nServer : - socket binding done.");

listen(socket\_desc, 3);

printf("\nServer : - Waiting for connections...");

c = sizeof(struct sockaddr\_in);

client\_sock=accept(socket\_desc,(struct sockaddr \*)&client,(socklen\_t\*)&c);

if(client\_sock < 0){

printf("\nServer : - Connection request from client rejected. Exiting.");

return 1;

}

printf("\nServer : - Connnection request from client accepted.");

memset(client\_message, '\0', sizeof client\_message);

memset(message, '\0', sizeof message);

if(recv(client\_sock,client\_message,200,0)<0){

printf("\nServer : - Data uplink from client failed.");

return 1;

}

printf("\nServer : - Password sent from client : %s",client\_message);

if(strcmp(pMessage,client\_message)==0)

strcpy(message,"Correct Password. Authenticated sucessfully..");

else strcpy(message,"Wrong Password!!!");

if(send(client\_sock,message,strlen(message),0)<0){

printf("\nServer : - Data downlink from server failed.");return 1; }

if(strcmp(message,"Wrong Password!!!")==0){

printf("\nAuthentication failed. Exiting..");return 1;}

memset(client\_message, '\0', sizeof client\_message);

memset(message, '\0', sizeof message);

printf("\nChat with client");

if(recv(client\_sock,client\_message,200,0)<0){

printf("\nServer : - Data uplink from client failed.");

return 1;

}

printf("\nClient : - %s",client\_message);

strcpy(message,"Hello there.");

if(send(client\_sock,message,strlen(message),0)<0){

printf("\nServer : - Data downlink from server failed.");return 1; }

printf("\nServer : - %s",message);

close(client\_sock);

return 0;

}

**Test Run[1]:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ gcc AuthenticateC.c -o C

nitish89@ADMINRG-R5IIR8M ~

$ gcc AuthenticateS.c -o S

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket

Server : - socket binding done.

Server : - Waiting for connections...

Server : - Connnection request from client accepted.

Server : - Password sent from client : nitish

Chat with client

Client : - Hi

Server : - Hello there.

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Socket created successfully.

Client : - Successfully connected with socket.

Client : - Enter password to authenticate : nitish

Sever : - Correct Password. Authenticated sucessfully..

Chat with server

Client : - Enter message : Hi

Server : - Hello there.

**Test Run[2]:**

**Terminal[1]**

nitish89@ADMINRG-R5IIR8M ~

$ ./S

Server : - Creating Socket

Server : - socket binding done.

Server : - Waiting for connections...

Server : - Connnection request from client accepted.

Server : - Password sent from client : nibedita

Authentication failed. Exiting..

**Terminal[2]**

nitish89@ADMINRG-R5IIR8M ~

$ ./C

Client : - Creating Socket

Client : - Successfully connected with socket.

Client : - Enter password to authenticate : nibedita

Sever : - Wrong Password!!!