**EXERCISE: 1**

**01-08-2019**

**AIM:** To create client - server architecture for reversing the string using TCP and UDP in C language.

**ALGORITHM:**

**TCP (Transmission Control Protocol)**

**Server:**

1.Create a socket file descriptor.

int sockfd = socket(domain, type, protocol)

2. Configure settings in address struct.

3. Bind the socket to the address and port number specified in address in the server program.

int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

4. Put the server socket in passive mode.

int listen(int sockfd, int backlog);

5. Read string sent by client by read system call.

6. Reverse the string.

7. Send reversed string to client by send system call.

**Client:**

1.Create a socket file descriptor.

int sockfd = socket(domain, type, protocol)

2. Configure settings in address struct.

3. Convert IPv4 addresses from text to binary form.

4. Connect the socket.

int connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

5. Send string to server side by send system call.

6. Read string sent by server by read system call.

**UDP (User Datagram Protocol)**

**Server:**

1.Create a socket file descriptor.

int sockfd = socket(domain, type, protocol)

2. Configure settings in address struct.

3. Bind the socket to the address and port number specified in address in the server program.

int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

4. Try to receive any incoming UDP datagram.

5. Reverse the string.

7. Send reversed string to client.

**Client:**

1.Create a socket file descriptor.

int sockfd = socket(domain, type, protocol)

2. Configure settings in address struct.

3.Send string to server side.

4.Receive string sent by server.

**CODE:**

**TCP Server:**

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#include<ctype.h>

#define PORT 8090

int main()

{

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

char str[100];

int addrlen = sizeof(address);

char buffer[1024] = { 0 };

char\* hello = "Hello from server";

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

if (bind(server\_fd, (struct sockaddr\*)&address, sizeof(address)) < 0) {

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0) {

perror("listen");

exit(EXIT\_FAILURE);

}

if ((new\_socket = accept(server\_fd, (struct sockaddr\*)&address, (socklen\_t\*)&addrlen)) < 0) {

perror("accept");

exit(EXIT\_FAILURE);

}

valread = read(new\_socket, str, sizeof(str));

int i, j, temp;

int l = strlen(str);

if(!isalnum(str[0]))

printf("\n Enter valid string ");

else

{

printf("\nString sent by client:%s\n", str);

for (i = 0, j = l - 1; i < j; i++, j--) {

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

send(new\_socket, str, sizeof(str), 0);

printf("\nModified string sent to client\n");

}

return 0;

}

**TCP Client:**

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 8090

int main()

{

struct sockaddr\_in address;

int sock = 0, valread;

struct sockaddr\_in serv\_addr;

char str[100];

printf("\nInput the string:\n");

scanf("%[^\n]s", str);

char buffer[1024] = { 0 };

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0))

< 0) {

printf("\n Socket creation error \n");

return -1;

}

memset(&serv\_addr, '0', sizeof(serv\_addr));

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

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

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("\nAddress not supported \n");

return -1;

}

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

printf("\nConnection Failed \n");

return -1;

}

int l = strlen(str);

send(sock, str, sizeof(str), 0);

valread = read(sock, str, l);

if(isalnum(str[0]))

printf("%s\n", str);

return 0;

}

**UDP Server:**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int main(){

int udpSocket, nBytes;

char buffer[1024];

struct sockaddr\_in serverAddr, clientAddr;

struct sockaddr\_storage serverStorage;

socklen\_t addr\_size, client\_addr\_size;

int i,j,temp;

udpSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(7891);

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

memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

bind(udpSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

addr\_size = sizeof serverStorage;

while(1){

nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr \*)&serverStorage, &addr\_size);

int l=strlen(buffer);

for (i = 0, j = l - 1; i < j; i++, j--) {

temp = buffer[i];

buffer[i] = buffer[j];

buffer[j] = temp;

}

sendto(udpSocket,buffer,nBytes,0,(struct sockaddr \*)&serverStorage,addr\_size);

}

return 0;

}

**UDP Client:**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

int main(){

int clientSocket, portNum, nBytes;

char buffer[1024];

struct sockaddr\_in serverAddr;

socklen\_t addr\_size;

clientSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(7891);

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

memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

addr\_size = sizeof serverAddr;

while(1){

printf("Input: ");

fgets(buffer,1024,stdin);

nBytes = strlen(buffer) + 1;

sendto(clientSocket,buffer,nBytes,0,(struct sockaddr \*)&serverAddr,addr\_size);

nBytes = recvfrom(clientSocket,buffer,1024,0,NULL, NULL);

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

}

return 0;

}

**OUTPUT:**

**TCP Server:**

[106117107@localhost network]$ gcc TCPserverReverse.c

[106117107@localhost network]$ ./a.out

Enter valid string

**TCP Client:**

[106117107@localhost network]$ gcc TCPreverse.c

[106117107@localhost network]$ ./a.out

Input the string:

**TCP Server :**

[106117107@localhost network]$ gcc TCPserverReverse.c

[106117107@localhost network]$ ./a.out

String sent by client:hello people

Modified string sent to client

**TCP Client :**

[106117107@localhost network]$ gcc TCPreverse.c

[106117107@localhost network]$ ./a.out

Input the string:

hello people

elpoep olleh

**UDP Server:**

[106117107@localhost network]$ gcc UDPserverReverse.c

[106117107@localhost network]$ ./a.out

**UDP Client:**

[106117107@localhost network]$ gcc UDPreverse.c

[106117107@localhost network]$ ./a.out

Input: Hello world

From server:

dlrow olleH

Input: computer

From server:

retupmoc

Input: end

From server:

dne

**RESULT**: A client - server architecture for reversing the string using TCP and UDP is implemented using C language.