Experiment – 6

server.c

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define PORT 4444

char\* crc(char \*input, char \*out){

int i,j,keylen,msglen, outlen;

char key[30],temp[30],quot[100],rem[30],key1[30];

printf("Enter Key: ");

fgets(key,30,stdin);

keylen=strlen(key);

msglen=strlen(input);

strcpy(key1,key);

for (i=0;i<keylen-1;i++) {

input[msglen+i]='0';}

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

temp[i]=input[i];

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

quot[i]=temp[0];

if(quot[i]=='0')

for (j=0;j<keylen;j++)

key[j]='0'; else

for (j=0;j<keylen;j++)

key[j]=key1[j];

for (j=keylen-1;j>0;j--) {

if(temp[j]==key[j])

rem[j-1]='0'; else

rem[j-1]='1';}

rem[keylen-1]=input[i+keylen];

strcpy(temp,rem);}

strcpy(rem,temp);

j=0;

for (i=0;i<msglen-1;i++){

out[j]=input[i];

j++;}

for (i=0;i<keylen-1;i++){

out[j]=rem[i];

j++;}

outlen=strlen(out);

printf("Final data is: ");

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

printf("%c", out[i]);

return out;}

int main(){char buffer[1024], out[130], \*ptr;

// Server socket id

int sockfd, ret, clientSocket;

// Server socket address structures

struct sockaddr\_in serverAddr;

// Client socket address structures

struct sockaddr\_in cliAddr;

// Stores byte size of server socket address

socklen\_t addr\_size;

// Child process id

pid\_t childpid;

// Creates a TCP socket id from IPV4 family

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

if (sockfd < 0) {

printf("Error in connection.\n");

exit(1);}

printf("Server Socket is created.\n");

// Initializing address structure with NULL

memset(&serverAddr, '\0',sizeof(serverAddr));

// Assign port number and IP address to the socket created

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(PORT);

// 127.0.0.1 is a loopback address

serverAddr.sin\_addr.s\_addr

= inet\_addr("127.0.0.1");

// Binding the socket id with the socket structure

ret = bind(sockfd,

(struct sockaddr\*)&serverAddr,

sizeof(serverAddr));

if (ret < 0){printf("Error in binding.\n");exit(1);}

// Listening for connections (upto 10)

if (listen(sockfd, 10) == 0) {

printf("Listening...\n\n");}int cnt = 0;

while (1) {

// Accept clients andstore their information in cliAddr

clientSocket = accept(sockfd,(struct sockaddr\*)&cliAddr, &addr\_size);

if (clientSocket < 0) {exit(1);}

// Displaying information of connected client

printf("Connection accepted from %s:%d\n",

inet\_ntoa(cliAddr.sin\_addr),

ntohs(cliAddr.sin\_port));

// Print number of clients connected till now

printf("Clients connected: %d\n\n", ++cnt);

// Creates a child process

if ((childpid = fork()) == 0) {

// Closing the server socket id

if(recv(clientSocket, buffer, 1024, 0)< 0) {

printf("Error in receiving data.\n");}

ptr = crc(buffer, out);

close(sockfd);

// Send a confirmation message to the client

send(clientSocket, out,strlen(out), 0);}}

close(clientSocket);

return 0;}

client.c

// Client Side program to test

// the TCP server that returns

// a 'hi client' message

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

// PORT number

#define PORT 4444

int main(){

int clientSocket, ret;

char input[100];

struct sockaddr\_in cliAddr;

struct sockaddr\_in serverAddr;

char buffer[1024];

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

if (clientSocket < 0) {

printf("Error in connection.\n");

exit(1);}

printf("Client Socket is created.\n");

// Initializing socket structure with NULL

memset(&cliAddr, '\0', sizeof(cliAddr));

// Initializing buffer array with NULL

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

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

// Assigning port number and IP address

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(PORT);

// 127.0.0.1 is Loopback IP

serverAddr.sin\_addr.s\_addr

= inet\_addr("127.0.0.1");

ret = connect(clientSocket,struct sockaddr\*)&serverAddr,

sizeof(serverAddr));

if (ret < 0) {

printf("Error in connection.\n");

exit(1);}

printf("Connected to Server.\n");

printf(" Enter the data\n");

fgets(input,100,stdin);

send(clientSocket, input, strlen(input), 0);

printf("data: %s",input);

if(recv(clientSocket, buffer, 1024, 0)< 0) {

printf("Error in receiving data.\n");}

// Printing the message on screen

else {

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

bzero(buffer, sizeof(buffer));

}return 0;}

