Using TCP/IP sockets, write a client-server program to make client send the file name and the server to send back the contents of the requested file name “sample.txt” with the following contents: “Hello we are at Computer Networks Lab”. Display suitable error message in case the file is not present in the server.

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <arpa/inet.h>

#define bufsize 1024

int main(){

int clientSocket;

char buffer[1024],fname[255];

struct sockaddr\_in serverAddr;

socklen\_t addr\_size;

/\*---- Create the socket. The three arguments are: ----\*/

/\* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) \*/

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

/\*---- Configure settings of the server address struct ----\*/

/\* Address family = Internet \*/

serverAddr.sin\_family = AF\_INET;

/\* Set port number, using htons function to use proper byte order \*/

serverAddr.sin\_port = htons(7891);

/\* Set IP address to localhost \*/

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

/\* Set all bits of the padding field to 0 \*/

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

/\*---- Connect the socket to the server using the address struct ----\*/

addr\_size = sizeof serverAddr;

connect(clientSocket, (struct sockaddr \*) &serverAddr, addr\_size);

printf("\nEnter filename: ");scanf("%s",fname);

send(clientSocket,fname,255,0);

printf("\nResponse:\n");

/\*---- Read the message from the server into the buffer ----\*/

while((recv(clientSocket, buffer, bufsize, 0))>0)

printf("%s",buffer);

printf("\n");

return close(clientSocket);

}

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <fcntl.h>

#define bufsize 1024

int main(){

int serverSocket, newSocket;

char buffer[bufsize];

char fname[255];

int fd,n;

struct sockaddr\_in serverAddr;

struct sockaddr\_storage serverStorage;

socklen\_t addr\_size;

/\*---- Create the socket. The three arguments are: ----\*/

/\* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) \*/

serverSocket = socket(PF\_INET, SOCK\_STREAM, 0);

/\*---- Configure settings of the server address struct ----\*/

/\* Address family = Internet \*/

serverAddr.sin\_family = AF\_INET;

/\* Set port number, using htons function to use proper byte order \*/

serverAddr.sin\_port = htons(7891);

/\* Set IP address to localhost \*/

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

/\* Set all bits of the padding field to 0 \*/

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

/\*---- Bind the address struct to the socket ----\*/

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

/\*---- Listen on the socket, with 5 max connection requests queued ----\*/

if(listen(serverSocket,5)==0)

printf("Listening\n");

else

printf("Error\n");

/\*---- Accept call creates a new socket for the incoming connection ---\*/

addr\_size = sizeof serverStorage;

newSocket = accept(serverSocket, (struct sockaddr \*) &serverStorage, &addr\_size);

/\*---- receive file name from the incoming connection ----\*/

recv(newSocket,fname,255,0);

fd=open(fname,O\_RDONLY);

if(fd==-1){

strcpy(buffer,"No file found!");

n = strlen(buffer);

}

else{

n=read(fd,buffer,bufsize);

}

send(newSocket,buffer,n,0);

close(newSocket);

return close(serverSocket);

}

Run server.c first

Run Server

Create Sample.txt

**Data transfer in unreliable network code using CRC (16-bits) Technique.**

#include<stdio.h>

#include<string.h>

#define N strlen(g)

char t[28];//stores test data

char cs[28]; //stores crc checksum

char g[]="10001000000100001"; //generator polynomial

int a,e,c;

void xor()

{

for(c = 1;c < N; c++) // do xor operation from index 1 till N-1

cs[c] = (( cs[c] == g[c])?'0':'1');

}

void crc()

{

for(e=0;e<N;e++)

cs[e]=t[e]; //copy the padded data to cs array

do{

if(cs[0]=='1') // if the first element is 1 , only then go ahead with xor operation

xor(); //xor with generator polynomial

for(c=0;c<N-1;c++)

cs[c]=cs[c+1]; // as we are not doing xor operation for the 0th index , we just push it out from the existing cs array and make the 1st index as the 0th index.

cs[c]=t[e++]; // as we pushed the element at the 0th index out , we take the next element from the padded data to our xor operand.

} while(e<=a+N-1); // we continue this process till all the extra elements from the padded data are added to our xor operand and the division process is complete.

}

int main()

{

printf("\nEnter data : ");

scanf("%s",t);

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

printf("\nGeneratng polynomial : %s",g);

a=strlen(t);

for(e=a;e<a+N-1;e++)

t[e]='0';

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

printf("\nPadded data is : %s",t);

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

crc();

printf("\nChecksum is : %s",cs);

for(e=a;e<a+N-1;e++)

t[e]=cs[e-a];

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

printf("\nFinal codeword is : %s",t);

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

printf("\nTest error detection 0(yes) 1(no)? : ");

scanf("%d",&e);

if(e==0)

{

do{

printf("\nEnter the position where error is to be inserted : ");

scanf("%d",&e);

}while(e==0 || e>a+N-1);

t[e-1]=(t[e-1]=='0')?'1':'0';

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

printf("\nErroneous data : %s\n",t);

}

crc();

for(e=0;(e<N-1) && (cs[e]!='1');e++){} // checksum (i.e cs array ) should have all zeroes after performing crc operation, so if any 1 is found e is not incremented and hence e will become less than N-1 and therefore we can say there is an error in the transmitted message.

if(e<N-1)

printf("\nError detected\n\n");

else

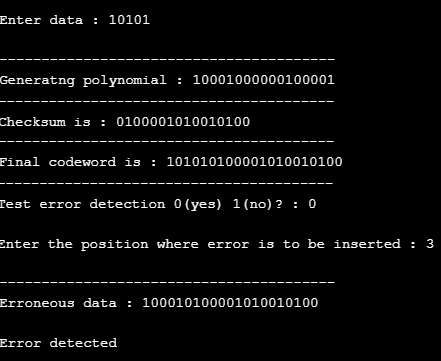
printf("\nNo error detected\n\n");

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

return 0;

}

**OUTPUT:**



#include <iostream>

#include <stdio.h> // Needed for printf() to get nice hex values

#include <stdlib.h> // Needed for rand()

using namespace std;

//----- Type defines ----------------------------------------------

typedef unsigned char byte; // Byte is a char

typedef unsigned short int word16; // 16-bit word is a short int

typedef unsigned int word32; // 32-bit word is an int

//----- Globals --------------------------------------------------

int BUFFER\_LEN=4096; // Length of buffer when using random data

//----- Prototypes -----------------------------------------------

word16 checksum(byte \*addr, word32 count,word32 initsum = 0);

//===== Main function=============================================

int main(void)

{

byte buff[BUFFER\_LEN]; // Buffer of packet bytes

word16 check; // 16-bit checksum value

word32 i,ch,v=0; // Loop counter and choice

printf("Use random values(0) or enter data(1) for checksum demo:");

cin>>ch;

if(ch){

printf("Enter the number of bytes:");

cin>>BUFFER\_LEN;

printf("Enter the %d bytes:",BUFFER\_LEN);

for (i=0; i<BUFFER\_LEN; i++)

cin>>buff[i];

v = 1;//indicates verbose

}

else

{

srand(time(NULL));

// Load buffer with BUFFER\_LEN random bytes

for (i=0; i<BUFFER\_LEN; i++)

buff[i] = (byte) rand();

}

if(v) printf("\nOriginal Data:%s\n",buff);

// Compute the 16-bit checksum

check = checksum(buff, BUFFER\_LEN);

printf("\nOriginal checksum of data = %04X\n", check);

printf("\nTest error detection 0(yes) 1(no)? : ");

cin>>ch;

if(ch==0)

{

do{

do{

printf("\nEnter the position where error is to be inserted(or 0 to stop) : ");

cin>>ch;

}while(ch>BUFFER\_LEN-1);

buff[ch]++;

}while(ch!=0);

}

if(v) printf("\nNew Data:%s\n",buff);

check = checksum(buff,BUFFER\_LEN,check);

// Output the checksum

printf("\nValidated checksum = %04X\n", check);

if(check)

printf("\nError Detected!\n");

else

printf("\nNo errors occured!\n");

}

//=====================================================================

// Compute Internet Checksum for count bytes beginning at location addr

// Passing initsum in case a checksum needs to be validated.

//=========================================================================

word16 checksum(byte \*addr, word32 count,word32 initsum)

{

word32 sum = initsum;

// Main summing loop

while(count > 1)

{

sum = sum + \*((word16 \*) addr);

count = count - 2;

(word16 \*) addr++;

}

// Add left-over byte, if any

if (count > 0)

sum = sum + \*((byte \*) addr);

// Fold 32-bit sum to 16 bits

while (sum>>16)

sum = (sum & 0xFFFF) + (sum >> 16);

return(~sum);

}