**CN Lab Session 1: Socket Programming**

1.) Write a UDP client-server program where client sends rows of a matrix to the server combines them together as a two dimensional matrix and  display the same.

**//Server**

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#define PORT 5000

#define MAXLINE 1000

int main()

{

char buffer[100];

int servsockfd, len,n;

struct sockaddr\_in servaddr, cliaddr;

bzero(&servaddr, sizeof(servaddr));

// Create a UDP Socket

servsockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// bind server address to socket descriptor

bind(servsockfd, (struct sockaddr\*)&servaddr, sizeof(servaddr));

//receive the datagram

len = sizeof(cliaddr);

n = recvfrom(servsockfd, buffer, sizeof(buffer),0, (struct sockaddr\*)&cliaddr,&len);

buffer[n] = '\0';

puts(buffer);

//Echoing back to the client

sendto(servsockfd, buffer, n, 0, (struct sockaddr\*)&cliaddr, sizeof(cliaddr));

getchar();

// close the descriptor

//close(servsockfd);

return 0;

}

**//Client**

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <stdlib.h>

#define PORT 5000

#define MAXLINE 1000

int main()

{

char buffer[100];

//char \*message = "Hello Server";

int sockfd, n,len;

struct sockaddr\_in servaddr, cliaddr;

//using a square matrix of 3\*2

int row=0;

printf("Enter the elements of the first row\n");

int a ,b, c;

scanf("%d %d %d",&a,&b, &c);

printf("Enter the elements of the second row \n");

int d ,e, f;

scanf("%d %d",&d ,&e, &f);

// clear servaddr

bzero(&servaddr, sizeof(servaddr));

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

char message[5];

message[0]=a+'0';

message[1]=b+'0';

message[2]=c+'0';

message[3]=d+'0';

message[4]=e+'0';

// create datagram socket

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

sendto(sockfd, message, MAXLINE, 0, (struct sockaddr\*)&servaddr, sizeof(servaddr));

len=sizeof(cliaddr);

// waiting for response

n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr\*)&cliaddr,&len );

buffer[n]='\0';

printf("Message from Server is \n");

//Just chck if it gives correct outpur or not , connection is already esablished

for(int i=0;i<3;i++)

{ //hardcoded till 3

printf("The %d th element of the matrix is :- %d \n",i,buffer[i]);

}

getchar();

// close the descriptor

close(sockfd);

return 0;

}

**Output**

**Graphical user interface, text, chat or text message

Description automatically generated**

2.) Write a TCP client which sends a string to a server program. Server displays the string along with client IP and ephemeral port number. Server then responds to the client by echoing back the string in uppercase. The process continues until one of them types “QUIT”.

**//Server**

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX 50

void servfunc(int conn\_fd,struct sockaddr\_in client\_address)

{

//display client ip and port number

//echo back received string in upper case

//if quit is received from client/server, end program

char buff[MAX];

int n=0;

char\* ip\_add=inet\_ntoa(client\_address.sin\_addr);

int port=client\_address.sin\_port;

printf("Client ip:%s Client port:%d \n",ip\_add,port);

while(1)

{

printf("WAITING from client\n");

//while(n==0)

memset(buff,0,sizeof(buff));

n = read(conn\_fd,buff,sizeof(buff));

buff[n]='\n';

printf("Client ip:%s Client port:%d and msg recieved is %s \n",ip\_add,port,buff);

if(strcmp("quit",buff)==0)

{

printf("server is closing..closed\n");

return;

}

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

{

buff[i]=toupper(buff[i]);

}

write(conn\_fd,buff,sizeof(buff));

}

}

int main()

{

int server\_sockfd, conn\_sockfd;

int server\_len,client\_len;

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

//create a socket for the server

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

//name the server socket

server\_address.sin\_family=AF\_INET;

//inet\_addr converts to unsigned long,

//else use htonl(INADDR\_ANY)

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

server\_address.sin\_port=htons(7280);

server\_len=sizeof(server\_address);

if(bind(server\_sockfd,(struct sockaddr\*)&server\_address,server\_len)!=0)

{

printf("socket binding failed\n");

exit(0);

}

else

{

printf("socked bound successfully\n");

}

//create a connection queue and wait for clients

if(listen(server\_sockfd,2)!=0)

{

printf("listen failed\n");

exit(0);

}

else

{

printf("server listening\n");

}

client\_len=sizeof(client\_address);

//when accepted a new client, a new socketfd is created

conn\_sockfd=accept(server\_sockfd,(struct sockaddr\*)&client\_address,&client\_len);

if(conn\_sockfd<0)

{

printf("server accept failed\n");

exit(0);

}

else

{

printf("server accepted the client\n");

}

servfunc(conn\_sockfd,client\_address);

close(server\_sockfd);

return 0;

}

**//Client**

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX 50

void clifunc(int sockfd)

{

//send string

//print returned string

//if returned string is quit, quit

char buff[MAX];

int n=0,recv\_len=0;

while(1)

{

memset(buff,0,sizeof(buff));

printf("Type message\n");

//while((buff[n++]=getchar())!='\n');

scanf("%s",buff);

write(sockfd,buff,sizeof(buff));

if(strcmp("quit",buff)==0)

{

printf("client closing\n");

return;

}

memset(buff,0,sizeof(buff));

n=read(sockfd,buff,sizeof(buff));

buff[n]='\n';

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

}

}

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

{

int sockfd;

int len;

struct sockaddr\_in server\_address;

int result;

char ch;

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

server\_address.sin\_family=AF\_INET;

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

server\_address.sin\_port=htons(7280);

len=sizeof(server\_address);

result=connect(sockfd,(struct sockaddr\*)&server\_address,len);

if(result == -1)

{

printf("connection error\n");

exit(0);

}

clifunc(sockfd);

close(sockfd);

return 0;

}

**Output**

**Graphical user interface, text

Description automatically generated**

3.) Implement concurrent Remote Math Server To perform arithmetic operations in the server and display the result at the client. The client accepts two integers and an operator from the user and sends it to the server. The server will performs the operation on integers and sends result back to the client which is displayed in the client.

**//Client**

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <arpa/inet.h>

#include <string.h>

#include <stdlib.h>

#define MAXSIZE 150

#define PORT 5000

#define MAXLINE 1000

typedef struct obj

{

double a,b,r;

char op;

char ans[10];

}obj1,\*obj\_ptr;

int main()

{

int sockfd,retval;char ch;

int recedbytes,sentbytes, recans;

struct sockaddr\_in serveraddr;

obj\_ptr buffer = (obj\_ptr)malloc(sizeof(obj1));

obj\_ptr buffer1 = (obj\_ptr)malloc(sizeof(obj1));

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

if(sockfd==-1)

{

printf("\nSocket Creation Error");

}

printf("\nSocket ID : %d\n",sockfd);

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(PORT);

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

retval=connect(sockfd,(struct sockaddr\*)&serveraddr,sizeof(serveraddr));

if(retval==-1)

{

printf("Connection error");

}

do

{

printf("Do you want to request? Yes/Stop\n");

scanf("%c",&ch);

scanf("%[^\n]%\*c",(buffer->ans));

if(strcmp(buffer->ans,"stop")==0)

{

puts("Stopping");

sentbytes=send(sockfd,buffer,sizeof(buffer),0);

close(sockfd);

}

else

{

printf("Enter in form a op b : ");

scanf("%lf %c %lf",&buffer->a, &buffer->op, &buffer->b);

sentbytes=send(sockfd,buffer,sizeof(obj1),0);

recedbytes=recv(sockfd,buffer1,sizeof(obj1),0);

printf("Result is: %.2lf \n",buffer1->r);

}

}while(strcmp(buffer->ans, "stop") != 0);

return 0;

}

**//Server**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <ctype.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define MAXSIZE 150

#define PORT 5000

#define MAXLINE 1000

typedef struct obj

{

double a,b,r;

char op;

char ans[10];

}obj1,\*obj\_ptr;

int main()

{

int sockfd,newsockfd,retval;

socklen\_t actuallen;

int recedbytes,sentbytes, sentans;

struct sockaddr\_in serveraddr,clientaddr;

obj\_ptr buffer = (obj\_ptr)malloc(sizeof(obj1));

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

if(sockfd==-1)

{

printf("\nSocket creation error");

}

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(PORT);

serveraddr.sin\_addr.s\_addr=htons(INADDR\_ANY);

bind(sockfd,(struct sockaddr\*)&serveraddr,sizeof(serveraddr));

puts("Server Running");

listen(sockfd,1);

actuallen=sizeof(clientaddr);

newsockfd=accept(sockfd,(struct sockaddr\*)&clientaddr,&actuallen);

do

{

recv(newsockfd,buffer,sizeof(obj1),0);

if(strcmp(buffer->ans, "stop") == 0)

{

puts("Stopping");

close(sockfd);

close(newsockfd);

}

else

{

printf("Client [%s:%d] requested: %.2lf %c %.2lf\n", inet\_ntoa(clientaddr.sin\_addr), ntohs(clientaddr.sin\_port), buffer->a, buffer->op, buffer->b);

switch (buffer->op)

{

case '+': buffer->r = buffer->a + buffer->b;

break;

case '-': buffer->r = buffer->a - buffer->b;

break;

case '\*': buffer->r = buffer->a \* buffer->b;

break;

case '/': buffer->r = buffer->a / buffer->b;

break;

case '%': buffer->r = buffer->a / buffer->b;

break;

default:

break;

}

sentbytes = send(newsockfd,buffer,sizeof(obj1),0);

}

}while(strcmp(buffer->ans, "stop") != 0);

return 0;

}

**Output**

**Graphical user interface, text

Description automatically generated**

4.) DayTime Server: Where client sends request to time server to send current time. Server responds by sending the current time . [Hint: read man pages of asctime() and localtime()] . Display server process id at client side along with time.

**//Client**

#include <stdlib.h>

#include <time.h>

int main()

{

int sockfd;

int len;

struct sockaddr\_in address;

struct tm \* timeinfo;

int result;

char \*reply;

int hour,mins,sec,pid;

/\* Create a socket for the client. \*/

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

/\* Name the socket, as agreed with the server. \*/

address.sin\_family = AF\_INET;

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

address.sin\_port = 9734;

len = sizeof(address);

/\* Now connect our socket to the server's socket. \*/

result = connect(sockfd, (struct sockaddr \*)&address, len);

if(result == -1)

{

perror("oops: client2");

exit(1);

}

/\* We can now read/write via sockfd. \*/

printf(" Sending request to get the time\n");

read(sockfd, &hour , 1);

read(sockfd, &mins , 1);

read(sockfd, &sec , 1);

read(sockfd, &pid , 1);

printf("%d:%d:%d", hour, mins, sec);

printf(" The process id is: %d",pid);

close(sockfd);

exit(0);

return 0;

}

**//Server**

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

int main()

{

time\_t rawtime;

struct tm \* timeinfo;

char \*reply;

int server\_sockfd, client\_sockfd;

int server\_len, client\_len;

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

int hour,mins,sec,pid;

/\* Create an unnamed socket for the server. \*/

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

/\* Name the socket. \*/

server\_address.sin\_family = AF\_INET;

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

server\_address.sin\_port = 9734;

server\_len = sizeof(server\_address);

bind(server\_sockfd, (struct sockaddr \*)&server\_address, server\_len);

/\* Create a connection queue and wait for clients. \*/

listen(server\_sockfd, 5);

while(1)

{

char ch;

printf("server waiting\n");

/\* Accept a connection. \*/

client\_len = sizeof(client\_address);

client\_sockfd = accept(server\_sockfd, (struct sockaddr \*)&client\_address, &client\_len);

/\* We can now read/write to client on client\_sockfd. \*/

//char \*inet\_ntoa(client\_addr.sin\_addr);

char \* ip\_add =inet\_ntoa(client\_address.sin\_addr);

int port=client\_address.sin\_port;

printf("IP:%s PORT:%d\n", ip\_add,port);

//get the time

time ( &rawtime );

timeinfo = localtime ( &rawtime );

reply = asctime(timeinfo);

printf ( "The current date/time is: %s", reply );

hour = timeinfo->tm\_hour;

mins = timeinfo->tm\_min;

sec = timeinfo->tm\_sec;

pid = getpid();

write(client\_sockfd, &hour, 1);

write(client\_sockfd, &mins, 1);

write(client\_sockfd, &sec, 1);

write(client\_sockfd, &pid, 1);

//close(client\_sockfd);

}

return 0;}

**Output**

**Graphical user interface, text

Description automatically generated**