**Assignment - 1**

*(60 marks) & beyond deadline: 30 marks*

1. Write a program to run TCP client and server socket programs (client and server must run in different Mininet hosts) where -
2. Client first says "Hi" and in response server says "Hello". **(10 marks)**
3. Server will interpret command "calc expr" where "expr" will look like a+b, a-b, ab, a/b. calc command will operate as the calculator. Calc command will be sent by the client and server will compute and return the results. **(20 marks)**
4. Server will maintain a fruit store and multiple clients will do transactions with the server. **(30 marks)**
   1. Store will have initially 20 apples and 10 Kg mango. b. Client will send command "buy fruit name quantity"
   2. Server will show the client IP and port and will assign a unique id to the transaction. This id should be shown to the client
   3. Server will further deduct the quantities based on the client transactions. If buy 5 apples, it should show 15 apples remaining to you once you connect to the server i.e., remaining items quantity must be shown to the clients before transactions.
   4. If a requested quantity is not available, server must reply "quantity not available message to that client.

**Solution:**

1. server.c program

#include <stdio.h>

#include <sys/socket.h>

#include <unistd.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

int main(){

int welcomeSocket, newSocket;

char buffer[1024];

struct sockaddr\_in serverAddr;

struct sockaddr\_in 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) \*/

welcomeSocket = 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("10.0.0.1");

/\* 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(welcomeSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

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

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

printf("Server Listening on port %d\n", serverAddr.sin\_port);

else

printf("Error\n");

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

while(1) {

addr\_size = sizeof serverStorage;

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

/\*---- Identify clients like this. The following information of client are taken from client due to connect function ----\*/

//Change of the following information at client side can not be done. However, padding field may be changed and that may be tried

struct sockaddr\_in\* cliIP = (struct sockaddr\_in\*)&serverStorage;

struct in\_addr ipAddr = cliIP->sin\_addr;

char str[INET\_ADDRSTRLEN];

inet\_ntop(AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN);

char\* ID = cliIP->sin\_zero;

char str2[8];

inet\_ntop(AF\_INET, &ID, str2, 8);

int i;

for (i=0;i<8;i++){

printf("%c", serverStorage.sin\_zero[i]);

}

/\* ---- Receive message from client, if any ---- \*/

recv(newSocket, buffer, 1024, 0);

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

/\*---- Send message to the socket of the incoming connection ----\*/

strcpy(buffer, "Hello");

send(newSocket,buffer,23,0);

close(newSocket);

}

return 0;

}

**client.c program**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

int main(){

int clientSocket;

char buffer[1024];

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t addr\_size;

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

/\* 1) Internet domain 2) Stream socket 3) Dlefault 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("10.0.0.1");

/\* 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 ----\*/

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

/\* ---- Send something to server ---- \*/

strcpy(buffer, "Hi..");

send(clientSocket,buffer, 1024, 0);

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

recv(clientSocket, buffer, 1024, 0);

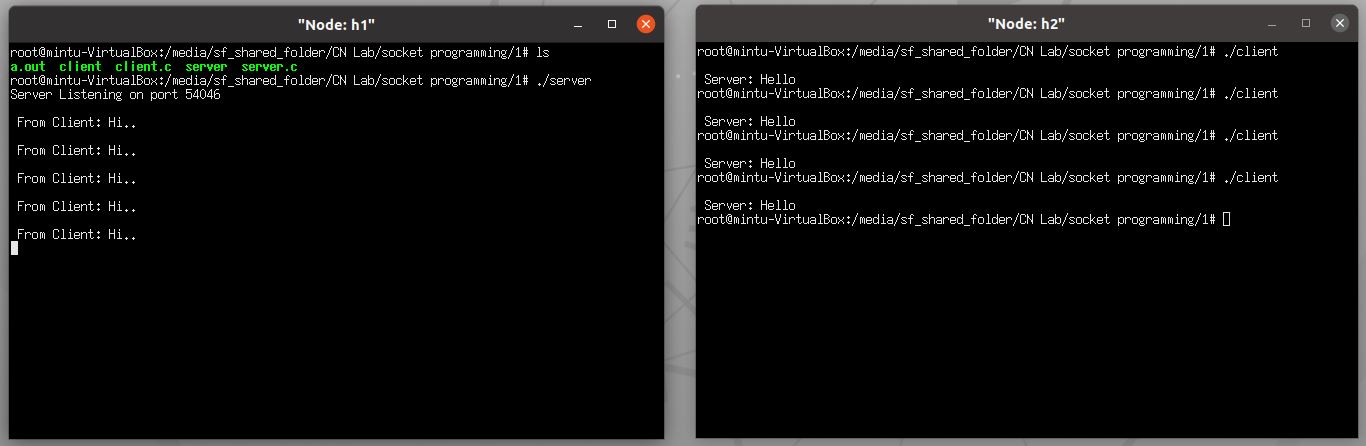
/\*---- Print the received message ----\*/

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

return 0;

}

*Program 1(a) output:*



(b) *server.c program*

#include <stdio.h>

#include <sys/socket.h>

#include <unistd.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

int calc(char ch[255]);

int main(){

int welcomeSocket, newSocket;

char buffer[1024];

int result;

struct sockaddr\_in serverAddr;

struct sockaddr\_in 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) \*/

welcomeSocket = 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("10.0.0.1");

/\* 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(welcomeSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

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

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

printf("Listening\n");

else

printf("Error\n");

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

while(1) {

addr\_size = sizeof serverStorage;

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

/\*---- Identify clients like this. The following information of client are taken from client due to connect function ----\*/

//Change of the following information at client side can not be done. However, padding field may be changed and that may be tried

struct sockaddr\_in\* cliIP = (struct sockaddr\_in\*)&serverStorage;

struct in\_addr ipAddr = cliIP->sin\_addr;

char str[INET\_ADDRSTRLEN]

inet\_ntop(AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN);

char\* ID = cliIP->sin\_zero;

char str2[8];

inet\_ntop(AF\_INET, &ID, str2, 8);

int i;

for (i=0;i<8;i++){

printf("%c", serverStorage.sin\_zero[i]);

}

/\* ---- Receive message from client, if any ---- \*/

recv(newSocket, buffer, 1024, 0);

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

/\*---- Send message to the socket of the incoming connection ----\*/

// strcpy(res, calc(buffer));

result = calc(buffer);

send(newSocket,&result,4,0);

close(newSocket);

}

return 0;

}

// function defs

int calc(char ch[255]){

int num1 = 0,num2 = 0;

char op;

int state = 0;

int result =0;

int i=0;

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

if (ch[i] != '\n'){

if (ch[i] >= '0' && ch[i] <= '9'){ //Checks if the character is a number

if (state == 0)

num1 = num1\*10 + ch[i] - '0'; // Convert ASCII to decimal

else

num2 = num2\*10 + ch[i] - '0'; // Convert ASCII to decimal

}

else{

op = ch[i] ;

state = 1;

}

}

}

switch(op)

{

case '+': result = num1 + num2;

break;

case '-': result = num1 - num2;

break;

case '\*': result = num1 \* num2;

break;

case '/': result = num1 / num2;

break;

}

return result;

}

*Client.c program*

// calc client

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

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

int clientSocket;

char buffer[1024];

int result;

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t addr\_size;

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

/\* 1) Internet domain 2) Stream socket 3) Dlefault 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("10.0.0.1");

/\* 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 ----\*/

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

/\* ---- Send something to server ---- \*/

strcpy(buffer, argv[2]);

send(clientSocket,buffer, 1024, 0);

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

recv(clientSocket, &result, 4, 0);

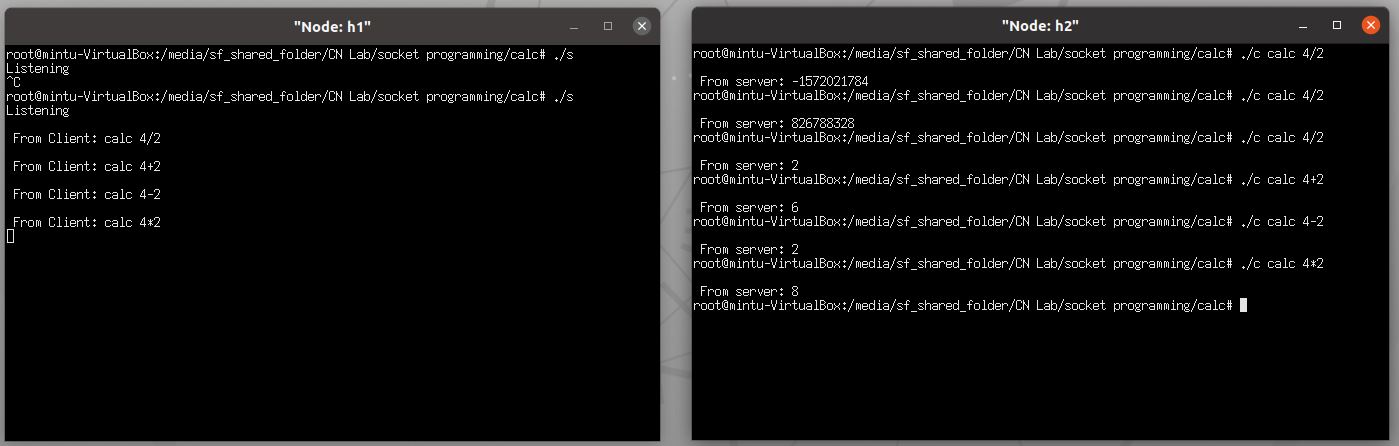
/\*---- Print the received message ----\*/

printf("\n From server: %d\n",result);

return 0;

}

Output 1(b):



(C)

Server.c code

#include <stdio.h>

#include <sys/socket.h>

#include <unistd.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#include <time.h>

int main()

int welcomeSocket, newSocket;

char buffer[1024];

int result;

int mango\_availble = 10;

int apple\_available = 20;

int txn\_id;

char \* fruit;

int qty;

struct sockaddr\_in serverAddr;

struct sockaddr\_in 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) \*/

welcomeSocket = 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("10.0.0.1");

/\* 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(welcomeSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

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

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

printf("Listening\n");

else

printf("Error\n");

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

while(1) {

addr\_size = sizeof serverStorage;

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

/\*---- Identify clients like this. The following information of client are taken from client due to connect function ----\*/

//Change of the following information at client side can not be done. However, padding field may be changed and that may be tried

struct sockaddr\_in\* cliIP = (struct sockaddr\_in\*)&serverStorage;

struct in\_addr ipAddr = cliIP->sin\_addr;

char str[INET\_ADDRSTRLEN];

inet\_ntop(AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN);

char\* ID = cliIP->sin\_zero;

char str2[8];

inet\_ntop(AF\_INET, &ID, str2, 8);

int i;

for (i=0;i<8;i++){

printf("%c", serverStorage.sin\_zero[i]);

}

/\*---- Send message to the socket of the incoming connection ----\*/

send(newSocket,&apple\_available,4,0);

send(newSocket,&mango\_availble,4,0);

/\* ---- Receive message from client, if any ---- \*/

recv(newSocket, buffer, 1024, 0);

recv(newSocket, fruit, 1024, 0);

recv(newSocket, &qty, 1024, 0);

printf("\nData received from client <%s, %d>: %s %s %d\n",inet\_ntoa(serverStorage.sin\_addr), serverStorage.sin\_port, buffer, fruit, qty);

if(strcmp(fruit, "apple") == 0){

if(apple\_available < qty){

strcpy(buffer, "Requested qty not available");

send(newSocket,buffer,1024,0);

}

else{

char txnId[30];

apple\_available = apple\_available - qty;

txn\_id = (int)time(NULL);

sprintf(buffer, "%d", txn\_id);

send(newSocket,buffer,1024,0);

}

}

if(strcmp(fruit, "mango") == 0){

if(mango\_availble < qty){

strcpy(buffer, "Requested qty not available");

send(newSocket,buffer,1024,0);

}

else{

mango\_availble = mango\_availble - qty;

txn\_id = (int)time(NULL);

sprintf(buffer, "%d", txn\_id);

send(newSocket,buffer,1024,0);

}

}

close(newSocket);

}

return 0;

}

client.c Program

// calc client

#include <stdio.h>

#include<stdlib.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

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

int clientSocket;

char buffer[1024];

int result;

int mango\_availble = -1;

int apple\_available = -1;

char\* fruit = argv[2];

int qty = atoi(argv[3]);

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t addr\_size;

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

/\* 1) Internet domain 2) Stream socket 3) Dlefault 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("10.0.0.1");

/\* 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 ----\*/

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

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

recv(clientSocket, &apple\_available, 4, 0);

recv(clientSocket, &mango\_availble, 4, 0);

printf("\nFrom server: ");

printf("\nMango available: %d \t Apple Available: %d\n", mango\_availble, apple\_available);

/\* ---- Send something to server ---- \*/

strcpy(buffer, argv[1]);

send(clientSocket,buffer, 1024, 0);

strcpy(buffer, argv[2]);

send(clientSocket,buffer, 1024, 0);

send(clientSocket,&qty, 4, 0);

// receive response fomr server

recv(clientSocket, buffer, 1024, 0);

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

return 0;

}

1(c) output:

