**Computer Networks**

**Laboratory**

**Experiment 3**

**Course code:** BCSE308P

**Slot :**

**Faculty :** N G Bhuwaeshwari ma’am

**Name :** Voladri Manohar Reddy

**Reg.no:** 21BRS1177

**Task : 1**

**Aim:**

To Write a client and server c program using TCP socket, where the server echoes the message which is sent by the client in reverse case (that is, if client says Hello, then server replies with hELLO), and also print the client’s IP address on server’s console.

**Server:**

**Algorithm:**

* Create a TCP socket using `socket()` function.
* Initialize the socket address structure with server IP address and port number using `bind()` function.
* Listen for incoming connections using `listen()` function.
* Accept incoming connection using `accept()` function and get client's IP address.
* Receive data from the client using `recv()` function.
* Convert the data received from the client to reverse case.
* Send the reverse case data back to the client using `send()` function.
* Close the connection using `close()` function

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

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

int server\_socket, client\_socket, read\_size;

struct sockaddr\_in server\_address, client\_address;

char client\_message[2000];

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

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

server\_address.sin\_addr.s\_addr = INADDR\_ANY;

server\_address.sin\_port = htons(5000);

bind(server\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address));

listen(server\_socket, 1);

puts("Waiting for incoming connections...");

while (1) {

socklen\_t client\_address\_len = sizeof(client\_address);

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

char client\_address\_str[INET\_ADDRSTRLEN];

inet\_ntop(AF\_INET, &client\_address.sin\_addr, client\_address\_str, INET\_ADDRSTRLEN);

printf("Connection from %s\n", client\_address\_str);

memset(client\_message, 0, sizeof(client\_message));

read\_size = recv(client\_socket, client\_message, 2000, 0);

printf("Received message from client: %s\n", client\_message);

for (int i = 0; i < read\_size; i++) {

if (client\_message[i] >= 'A' && client\_message[i] <= 'Z') {

client\_message[i] += 'a' - 'A';

} else if (client\_message[i] >= 'a' && client\_message[i] <= 'z') {

client\_message[i] -= 'a' - 'A';

}

}

send(client\_socket, client\_message, strlen(client\_message), 0);

printf("Sent message to client: %s\n", client\_message);

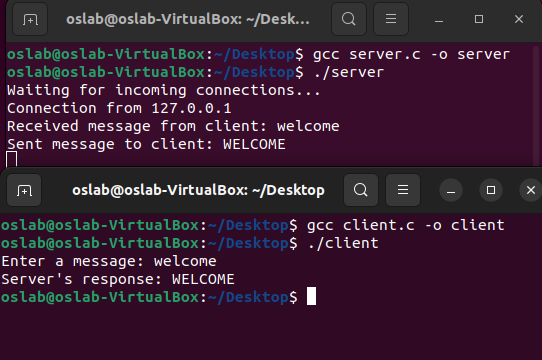
close(client\_socket);

}

return 0;

}

**Output:**



**Client :**

**Algorithms:**

* Create a TCP socket using `socket()` function.
* Initialize the socket address structure with server IP address and port number using `connect()` function.
* Prompt user to enter message.
* Send the message to server using `send()` function.
* Receive the reversed message from server using `recv()` function.
* Display the reversed message on the console.
* Close the socket using `close()` function.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

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

int client\_socket, read\_size;

struct sockaddr\_in server\_address;

char server\_response[2000], message[2000];

client\_socket = 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(5000);

connect(client\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address));

printf("Enter a message: ");

fgets(message, 2000, stdin);

message[strcspn(message, "\n")] = 0;

send(client\_socket, message, strlen(message), 0);

memset(server\_response, 0, sizeof(server\_response));

read\_size = recv(client\_socket, server\_response, 2000, 0);

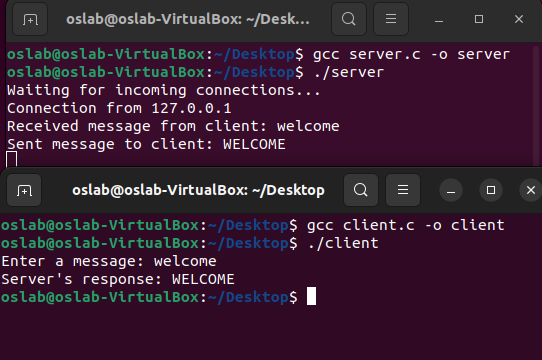
printf("Server's response: %s\n", server\_response);

close(client\_socket);

return 0;

}

**Output:**



**Task : 2**

**Aim:**

Implement a connection-oriented service to compute the grade of a student. The client has to send five subject marks to the server and the server computes the grade as follows: • If the percentage is greater than or equal to 90, then the grade is ‘S’ • If the percentage is greater than or equal to 80 and less than 90, then the grade is ‘A’ • If the percentage is greater than or equal to 70 and less than 80, then the grade is ‘B’ • If the percentage is greater than or equal to 60 and less than 70, then the grade is ‘C’ • If the percentage is greater than or equal to 50 and less than 60, then the grade is ‘D’ • If the percentage is less than 50, then the grade is ‘E’

**Server :**

**ALGORITHAM:**

* Create a TCP socket using `socket()` function.
* Initialize the socket address structure with server IP address and port number using `bind()` function.
* Listen for incoming connections using `listen()` function.
* Accept incoming connection using `accept()` function.
* Receive data from the client using `recv()` function.
* Compute the grade based on the received marks as follows:
  + Calculate the total marks by adding all the received marks.
  + Calculate the percentage by dividing the total marks by 500 and multiplying by 100.
  + Determine the grade based on the percentage using the rules mentioned in the problem statement.
* Send the grade back to the client using `send()` function.
* Close the connection using `close()` function.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 5000

#define MAX\_SUBJECTS 5

void calculate\_grade(int \*marks, char \*grade) {

int total\_marks = 0;

float percentage;

for (int i = 0; i < MAX\_SUBJECTS; i++) {

total\_marks += marks[i];

}

percentage = ((float)total\_marks / (MAX\_SUBJECTS \* 100)) \* 100;

if (percentage >= 90) {

\*grade = 'S';

} else if (percentage >= 80 && percentage < 90) {

\*grade = 'A';

} else if (percentage >= 70 && percentage < 80) {

\*grade = 'B';

} else if (percentage >= 60 && percentage < 70) {

\*grade = 'C';

} else if (percentage >= 50 && percentage < 60) {

\*grade = 'D';

} else {

\*grade = 'E';

}

}

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

int marks[MAX\_SUBJECTS];

char grade;

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

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the PORT

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))) {

perror("setsockopt");

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);

}

printf("Connection established with client\n");

while (1) {

if (recv(new\_socket, marks, sizeof(marks), 0) == -1) {

perror("recv");

exit(EXIT\_FAILURE);

}

calculate\_grade(marks, &grade);

if (send(new\_socket, &grade, sizeof(grade), 0) == -1) {

perror("send");

exit(EXIT\_FAILURE);

}

printf("Grade sent to client\n");

}

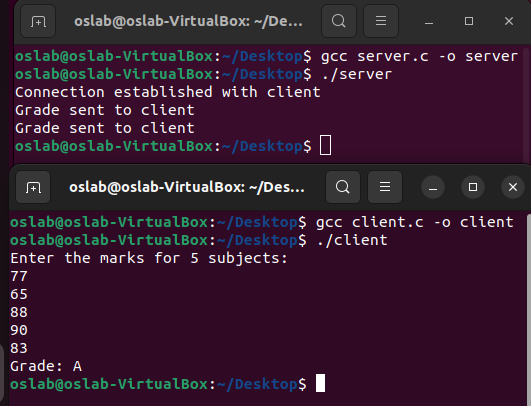
close(new\_socket);

close(server\_fd);

return 0;

}

**Output**:



**Client :**

**Algorithm:**

* Create a TCP socket using `socket()` function.
* Initialize the socket address structure with server IP address and port number using `connect()` function.
* Prompt user to enter five subject marks.
* Send the marks to server using `send()` function.
* Receive the grade from server using `recv()` function.
* Display the grade on the console.
* Close the socket using `close()` function

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 5000

#define MAX\_SUBJECTS 5

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

int sock = 0;

struct sockaddr\_in serv\_addr;

int marks[MAX\_SUBJECTS];

char grade;

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

perror("socket");

exit(EXIT\_FAILURE);

}

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) {

perror("inet\_pton");

exit(EXIT\_FAILURE);

}

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

perror("connect");

exit(EXIT\_FAILURE);

}

printf("Enter the marks for %d subjects:\n", MAX\_SUBJECTS);

for (int i = 0; i < MAX\_SUBJECTS; i++) {

scanf("%d", &marks[i]);

}

if (send(sock, marks, sizeof(marks), 0) == -1) {

perror("send");

exit(EXIT\_FAILURE);

}

if (recv(sock, &grade, sizeof(grade), 0) == -1) {

perror("recv");

exit(EXIT\_FAILURE);

}

printf("Grade: %c\n", grade);

close(sock);

return 0;

}

**Output:**

