ASSIGNMENT 7
16/10/24

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS7D
TOPIC : DISTRIBUTED SYSTEM
CODE : CS-17201

```c
//Q1 Client

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE] = {0};
    char input[BUFFER_SIZE];

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    if (connect(sock, (struct sockaddr *) &server_addr,
sizeof(server_addr)) < 0) {
        perror("Connection failed");
        close(sock);
        exit(EXIT_FAILURE);
    }

    printf("Enter a string: ");
    fgets(input, BUFFER_SIZE, stdin);
    input[strcspn(input, "\n")] = '\0';
    send(sock, input, strlen(input), 0);

    read(sock, buffer, BUFFER_SIZE);
    printf("Uppercase from server: %s\n", buffer);

    close(sock);
    return 0;
}
```

```c
//Q1 Server

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <arpa/inet.h>

#define BUFFER_SIZE 1024

void to_uppercase(char *str) {
    for (int i = 0; str[i]; i++) {
        str[i] = toupper(str[i]);
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int port = atoi(argv[1]);
    if (port <= 0) {
        fprintf(stderr, "Invalid port number.\n");
        exit(EXIT_FAILURE);
    }

    int server_fd, new_socket;
    struct sockaddr_in address;
    char buffer[BUFFER_SIZE] = {0};
    int addrlen = sizeof(address);

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0))
== 0) {
        perror("Socket failed");
        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");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d...\n", port);
    while (1) {
        new_socket = accept(server_fd, (struct sockaddr
*) &address, (socklen_t * ) & addrlen);
        if (new_socket < 0) {
            perror("Accept failed");
            continue;
        }

        read(new_socket, buffer, BUFFER_SIZE);
        printf("Received: %s\n", buffer);

        to_uppercase(buffer);

        send(new_socket, buffer, strlen(buffer), 0);
        printf("Sent: %s\n", buffer);

        close(new_socket);
    }

    close(server_fd);
    return 0;
}
```

```c
//Load Balancer
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define BUFFER_SIZE 1024

int get_cpu_load(const char *ip, int port) {
    return rand() % 100;
}

void forward_message(const char *ip, int port, const
char *message, char *response) {
    int sock;
    struct sockaddr_in server_address;
    char buffer[BUFFER_SIZE] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(port);
    if (inet_pton(AF_INET, ip,
&server_address.sin_addr) <= 0) {
        perror("Invalid address/Address not
supported");
        close(sock);
        exit(EXIT_FAILURE);
    }

    if (connect(sock, (struct sockaddr *)
&server_address, sizeof(server_address)) < 0) {
        perror("Connection to server failed");
        close(sock);
        exit(EXIT_FAILURE);
    }

    send(sock, message, strlen(message), 0);

    read(sock, buffer, BUFFER_SIZE);
    strcpy(response, buffer);
```

```
    close(sock);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <server1_port>
<server2_port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int server1_port = atoi(argv[1]);
    int server2_port = atoi(argv[2]);

    if (server1_port <= 0 || server2_port <= 0) {
        fprintf(stderr, "Invalid port numbers.\n");
        exit(EXIT_FAILURE);
    }

    int load_balancer_sock, client_sock;
    struct sockaddr_in load_balancer_address,
client_address;
    socklen_t client_address_len =
sizeof(client_address);
    char buffer[BUFFER_SIZE] = {0};
    const char *server_ip = "127.0.0.1";

    if ((load_balancer_sock = socket(AF_INET,
SOCK_STREAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    load_balancer_address.sin_family = AF_INET;
    load_balancer_address.sin_addr.s_addr = INADDR_ANY;
    load_balancer_address.sin_port = htons(8080);

    if (bind(load_balancer_sock, (struct sockaddr *)
&load_balancer_address, sizeof(load_balancer_address))
< 0) {
        perror("Bind failed");
        close(load_balancer_sock);
        exit(EXIT_FAILURE);
    }

    if (listen(load_balancer_sock, 5) < 0) {
```

```
        perror("Listen failed");
        close(load_balancer_sock);
        exit(EXIT_FAILURE);
    }

    printf("Load balancer listening on port
8080...\n");

    while (1) {
        client_sock = accept(load_balancer_sock,
(struct sockaddr *) &client_address,
&client_address_len);
        if (client_sock < 0) {
            perror("Accept failed");
            continue;
        }

        memset(buffer, 0, BUFFER_SIZE);
        read(client_sock, buffer, BUFFER_SIZE);
        printf("Received message from client: %s\n",
buffer);

        int cpu_load1 = get_cpu_load(server_ip,
server1_port);
        int cpu_load2 = get_cpu_load(server_ip,
server2_port);
        printf("CPU Load - Server 1: %d%%, Server 2:
%d%%\n", cpu_load1, cpu_load2);

        int selected_port = (cpu_load1 <= cpu_load2) ?
server1_port : server2_port;
        printf("Forwarding message to server on port
%d...\n", selected_port);

        char response[BUFFER_SIZE] = {0};
        forward_message(server_ip, selected_port,
buffer, response);
        send(client_sock, response, strlen(response),
0);
        close(client_sock);
    }

    close(load_balancer_sock);
    return 0;
}
```

./q1s 15552

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → clang q1s.c -o q1s
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → ./q1s 15551
Server listening on port 15551...
Received: hello world
Sent: HELLO WORLD
```

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → clang q1s.c -o q1s
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → ./q1s 15552
Server listening on port 15552...
Received: testing program
Sent: TESTING PROGRAM
```

.../sem7/dsys/2024-10-16

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → clang q2.c -o q2
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → ./q2 15551 15552
Load balancer listening on port 8080...
Received message from client: hello world
CPU Load - Server 1: 7%, Server 2: 49%
Forwarding message to server on port 15551...
Received message from client: testing program
CPU Load - Server 1: 73%, Server 2: 58%
Forwarding message to server on port 15552...
```

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → clang q1c.c -o q1c
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → ./q1c
Enter a string: hello world
Uppercase from server: HELLO WORLD
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → ./q1c
Enter a string: testing program
Uppercase from server: TESTING PROGRAM
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 →
```

```java
//Q2 Client

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class q3c {
    public q3c() {
    }

    public static void main(String[] var0) {
        try {
            Socket var1 = new Socket("localhost",
6789);

            try {
                System.out.println("Connected to the
server!");
                BufferedReader var2 = new
BufferedReader(new InputStreamReader(System.in));
                DataOutputStream var3 = new
DataOutputStream(var1.getOutputStream());
                BufferedReader var4 = new
BufferedReader(new
InputStreamReader(var1.getInputStream()));
                System.out.print("Enter a string: ");
                String var5 = var2.readLine();
                var3.writeBytes(var5 + "\n");
                String var6 = var4.readLine();
                System.out.println("Received from
server: " + var6);
            } catch (Throwable var8) {
                try {
                    var1.close();
                } catch (Throwable var7) {
                    var8.addSuppressed(var7);
                }

                throw var8;
            }

            var1.close();
        } catch (IOException var9) {
```

```
            var9.printStackTrace();
        }

    }
}

//Q2 Server

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

public class q3s {
    public q3s() {
    }

    public static void main(String[] var0) {
        try {
            ServerSocket var1 = new ServerSocket(6789);

            try {
                System.out.println("Server is waiting
for a client...");
                Socket var2 = var1.accept();
                System.out.println("Client
connected!");
                BufferedReader var3 = new
BufferedReader(new
InputStreamReader(var2.getInputStream()));
                DataOutputStream var4 = new
DataOutputStream(var2.getOutputStream());
                String var5 = var3.readLine();
                System.out.println("Received from
client: " + var5);
                String var6 = var5.toUpperCase();
                var4.writeBytes(var6 + "\n");
                System.out.println("Sent to client: " +
var6);
                var2.close();
            } catch (Throwable var8) {
                try {
                    var1.close();
```

```java
                } catch (Throwable var7) {
                    var8.addSuppressed(var7);
                }

                throw var8;
            }

            var1.close();
        } catch (IOException var9) {
            var9.printStackTrace();
        }

    }
}
```

```
.../sem7/dsys/2024-10-16
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → javac q3s.java
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → java q3s
Server is running and waiting for clients...
Client connected!
Received from client: hello world
Sent to client: HELLO WORLD
Client disconnected.
Client connected!
Received from client: testing program
Sent to client: TESTING PROGRAM
Client disconnected.
```

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → javac q3c.java
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → java q3c
Connected to the server!
Enter a string: hello world
Received from server: HELLO WORLD
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 → java q3c
Connected to the server!
Enter a string: testing program
Received from server: TESTING PROGRAM
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-16
 →
```