**Name-Ujjawal Mandhani**

**Batch-F6**

**Enroll no.-9918103237**

**ISL LAB WEEK #1**

**Server.c**

// Server side C/C++ program to demonstrate Socket programming

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

#define PORT 8080

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

{

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[1024] = {0};

char \*hello = "Hello from server";

// Creating socket file descriptor

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

{

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

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);

// Forcefully attaching socket to the port 8080

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);

}

valread = read(new\_socket, buffer, 1024);

printf("Encrypted Data : %s\n", buffer); //98

for (int i = 0; i < strlen(buffer); i++)

{

buffer[i] = buffer[i] - 3;

if (buffer[i] < 97)

{

buffer[i] = 122 - (96 - buffer[i]);

}

}

printf("After Decryption %s\n", buffer);

send(new\_socket, hello, strlen(hello), 0);

//printf("message sent\n");

return 0;

}

**Client.c**

// Client side C/C++ program to demonstrate Socket programming

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#define PORT 8080

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

{

int sock = 0, valread;

struct sockaddr\_in serv\_addr;

char \*hello = "Hello from client";

char buffer[1024] = {0};

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

{

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0)

{

printf("\nInvalid address/ Address not supported \n");

return -1;

}

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

{

printf("\nConnection Failed \n");

return -1;

}

int n;

scanf("%d", &n);

char msg[100];

char c;

scanf("%c", &c);

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

{

scanf("%c", &msg[i]);

}

for (int i = 0; i < strlen(msg); i++)

{

msg[i] = (msg[i] + 3);

if (msg[i] > 122)

{

msg[i] = 96 + msg[i] - 122;

}

}

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

printf("%s message sent\n", msg);

valread = read(sock, buffer, 1024);

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

return 0;

}

