**Module - 4 Assignment**

**Vishnu Sankar J**

1. **Explain the connection procedure followed in client server communication**

**Server.c**

#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 server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[BUFFER\_SIZE] = {0};

char \*hello = "Hello from server";

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

perror("socket failed");

exit(EXIT\_FAILURE);

}

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);

}

valread = read( new\_socket , buffer, BUFFER\_SIZE);

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

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

printf("Hello message sent\n");

return 0;

}

**Client.c:**

#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 argc, char const \*argv[]) {

int sock = 0, valread;

struct sockaddr\_in serv\_addr;

char \*hello = "Hello from client";

char buffer[BUFFER\_SIZE] = {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);

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;

}

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

printf("Hello message sent\n");

valread = read( sock , buffer, BUFFER\_SIZE);

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

return 0;

}

Sock\_stream: Provides a reliable, two-way, connection-based byte stream

AF\_INET: IPv4 protocol

1. **What is the use of bind() function in socket programming ?**

In socket programming, the bind () function is used to associate a socket with a specific network address, including the local address and port on which the server will listen for incoming connections.

int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

1. **What is Datagram Socket ?**

A datagram socket is a type of socket used in network programming that provides a connectionless, message-oriented communication between processes running on different hosts. Datagram sockets operate using the User Datagram Protocol (UDP) at the transport layer of the Internet Protocol Suite (TCP/IP). Unlike stream sockets (e.g., SOCK\_STREAM), which provide reliable, bi-directional, connection-oriented communication (like TCP), datagram sockets provide unreliable, connectionless, and packet-oriented communication.

1. **Write a server/client model socket program to exchange hello message between them.**

**Server.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8888

#define BUFFER\_SIZE 1024

int main() {

int server\_socket, client\_socket;

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

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

char buffer[BUFFER\_SIZE] = {0};

const char \*hello\_message = "Hello from server!";

if ((server\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

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

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

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

if (bind(server\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address)) == -1) {

perror("Bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_socket, 1) == -1) {

perror("Listen failed");

exit(EXIT\_FAILURE);

}

printf("Server listening on port %d\n", PORT);

if ((client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &client\_address\_len)) == -1) {

perror("Accept failed");

exit(EXIT\_FAILURE);

}

printf("Connection accepted from %s:%d\n", inet\_ntoa(client\_address.sin\_addr), ntohs(client\_address.sin\_port));

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

printf("Hello message sent to client\n");

read(client\_socket, buffer, BUFFER\_SIZE);

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

close(client\_socket);

close(server\_socket);

return 0;

}

**Client.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8888

#define BUFFER\_SIZE 1024

int main() {

int client\_socket;

struct sockaddr\_in server\_address;

char buffer[BUFFER\_SIZE] = {0};

const char \*hello\_message = "Hello from client!";

if ((client\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

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

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

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

if (connect(client\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address)) == -1) {

perror("Connection failed");

exit(EXIT\_FAILURE);

}

printf("Connected to server\n");

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

printf("Hello message sent to server\n");

read(client\_socket, buffer, BUFFER\_SIZE);

printf("Received message from server: %s\n", buffer);

close(client\_socket);

return 0;

}

1. **Write a TCP server-client program to check if a given string is Palindrome**

**Input: level**

**Output: Palindrome**

**Input: Assessment**

**Output: Not a Palindrome**

**Server.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8888

#define BUFFER\_SIZE 1024

int isPalindrome(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

if (str[i] != str[len - i - 1])

return 0;

}

return 1;

}

int main() {

int server\_socket, client\_socket;

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

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

char buffer[BUFFER\_SIZE] = {0};

if ((server\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

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

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

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

if (bind(server\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address)) == -1) {

perror("Bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_socket, 1) == -1) {

perror("Listen failed");

exit(EXIT\_FAILURE);

}

printf("Server listening on port %d\n", PORT);

if ((client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &client\_address\_len)) == -1) {

perror("Accept failed");

exit(EXIT\_FAILURE);

}

printf("Connection accepted from %s:%d\n", inet\_ntoa(client\_address.sin\_addr), ntohs(client\_address.sin\_port));

read(client\_socket, buffer, BUFFER\_SIZE);

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

int palindrome = isPalindrome(buffer);

if (palindrome)

send(client\_socket, "Palindrome", 11, 0);

else

send(client\_socket, "Not a Palindrome", 16, 0);

close(client\_socket);

close(server\_socket);

return 0;

}

**Client.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8888

#define BUFFER\_SIZE 1024

int main() {

int client\_socket;

struct sockaddr\_in server\_address;

char buffer[BUFFER\_SIZE] = {0};

if ((client\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

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

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

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

if (connect(client\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address)) == -1) {

perror("Connection failed");

exit(EXIT\_FAILURE);

}

printf("Connected to server\n");

printf("Enter a string: ");

fgets(buffer, BUFFER\_SIZE, stdin);

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

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

printf("String sent to server\n");

read(client\_socket, buffer, BUFFER\_SIZE);

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

close(client\_socket);

return 0;

}

1. **Write an example to demonstrate UDP server-client program**

**Server.c:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from server";

struct sockaddr\_in servaddr, cliaddr;

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(PORT);

if ( bind(sockfd, (const struct sockaddr \*)&servaddr,

sizeof(servaddr)) < 0 )

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

int len, n;

len = sizeof(cliaddr); //len is value/result

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, ( struct sockaddr \*) &cliaddr,

&len);

buffer[n] = '\0';

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

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &cliaddr,

len);

printf("Hello message sent.\n");

return 0;

}

**Client.c:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from client";

struct sockaddr\_in servaddr;

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(PORT);

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

int n, len;

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &servaddr,

sizeof(servaddr));

printf("Hello message sent.\n");

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, (struct sockaddr \*) &servaddr,

&len);

buffer[n] = '\0';

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

close(sockfd);

return 0;

}