

Date: 27 / 09 / 2022

Lab Practical #10:

Implement Client-Server Socket programming using C language

Practical Assignment #10:

- 1. Write a C code for TCP Server-Client Socket Programming.**
- 2. Write a C code for UDP Server-Client Socket Programming.**

1. For TCP Server-Client:

TCP Server C Program:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
```

```
// Function designed for chat between client and server.
```

```
void func(int connfd)
```

```
{
```

```
    char buff[MAX];
```

```
    int n;
```

```
    // infinite loop for chat
```

```
    for (;;) {
```

```
        bzero(buff, MAX);
```

```
        // read the message from client and copy it in buffer
```

Date: 27 / 09 / 2022

```
read(connfd, buff, sizeof(buff));
// print buffer which contains the client contents
printf("From client: %s\t To client : ", buff);
bzero(buff, MAX);
n = 0;
// copy server message in the buffer
while ((buff[n++] = getchar()) != '\n')
    ;

// and send that buffer to client
write(connfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and chat ended.
if (strncmp("exit", buff, 4) == 0) {
    printf("Server Exit...\n");
    break;
}
}
}

// Driver function
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

Date: 27 / 09 / 2022

```
if (sockfd == -1) {
    printf("socket creation failed...\n");
    exit(0);
}
else
    printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
    printf("socket bind failed...\n");
    exit(0);
}
else
    printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
    printf("Listen failed...\n");
    exit(0);
}
else
    printf("Server listening..\n");
```

Date: 27 / 09 / 2022

```
len = sizeof(cli);

// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
    printf("server accept failed...\n");
    exit(0);
}
else
    printf("server accept the client...\n");

// Function for chatting between client and server
func(connfd);

// After chatting close the socket
close(sockfd);
}
```

Date: 27 / 09 / 2022

TCP Client C Program:

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strcmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
        }
    }
}
```

Date: 27 / 09 / 2022

```
        break;
    }
}

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
        != 0) {
```

Date: 27 / 09 / 2022

```
printf("connection with the server failed...\n");
exit(0);
}
else
    printf("connected to the server..\n");

// function for chat
func(sockfd);

// close the socket
close(sockfd);
}
```

Date: 27 / 09 / 2022

OUTPUT :

Server Side :

```
Socket successfully created..  
Socket successfully binded..  
Server listening..  
server accept the client...  
From client: hi  
    To client : hello  
From client: exit  
    To client : exit  
Server Exit...
```

Client Side :

```
Socket successfully created..  
connected to the server..  
Enter the string : hi  
From Server : hello  
Enter the string : exit  
From Server : exit  
Client Exit...
```


Date: 27 / 09 / 2022

2. For UDP Server-Client:

UDP Server C Program:

```
// Server side implementation of UDP client-server model
#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

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
```

Date: 27 / 09 / 2022

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

// Filling server information
servaddr.sin_family = AF_INET; // IPv4
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);

// Bind the socket with the server address
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);
```

Date: 27 / 09 / 2022

```
printf("Hello message sent.\n");  
  
return 0;  
}
```

Date: 27 / 09 / 2022

UDP Client C Program:

```
// Client side implementation of UDP client-server model
#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

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
```

Date: 27 / 09 / 2022

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

// Filling server information
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;
}
```

Date: 27 / 09 / 2022

OUTPUT :

Server Side :

```
$ ./server  
Client : Hello from client  
Hello message sent.
```

Client Side :

```
$ ./client  
Hello message sent.  
Server : Hello from server
```