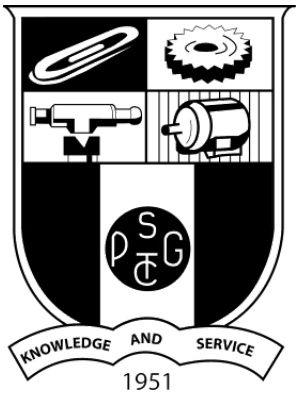
TCP Chat Program

19Z510 – COMPUTER NETWORKS LABORATORY

Anandkumar NS (22Z209)

BACHELOR OF ENGINEERING



Date: 18/08/2024

DEPARTMENT OF COMPUTER SCIENCE

ENGINEERING PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

Aim

To implement a simple TCP server in C that listens for incoming client connections, receives messages from clients, and responds with messages. The server should handle basic communication by reading data sent by a client, displaying it, and sending a response back. The server will also handle termination when the client sends an exit command.

Server

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#define MAX 80
#define PORT 3000
#define SA struct sockaddr

void func(int connfd)
{
   char buff[MAX];
   int n;

   for (;;) {
      bzero(buff, MAX);

      // read the message from client and copy it in buffer
      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 'Exit' then stop
      if (strncmp("exit", buff, 4) == 0) {
```

```
            printf("Server Exit...\n");
            break;
        }
    }
}

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

// af_inet is the ipv4 protocol for communication, AF_INET6 is the protocol for ipv6

//sock_stream is the TCP method that uses a method where a connection based protocol is enforced... When the connection is terminated  by one side, it is ended by both sides.
// the other option is sock_dgram which is a datagram based protocol where after one datagram is sent and a reply is recieved, then the connection terminates.

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
// if return is -1, then creation failed
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
//the bzero function erases the data in the n bytes of the memory starting at the pointer that is passed by writing '\0' in its location
    bzero(&servaddr, sizeof(servaddr));

//the below is how we cast the struct sockaddr_in* to struct      sock_addr*
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

//bind command accepts the socket file descriptor, 'const struct sockaddr *my_addr, suize of the address
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");

//listen () accpets the socket file desciptor and the backlog number ( the number of pendng connections)
```

```
    if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
    }
    else
        printf("Server listening..\n");
    len = sizeof(cli);

// accept() takes in the socket file descriptor and the pointer to the client address,
lenght of the address
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("server accept failed...\n");
        exit(0);
    }
    else
        printf("server accept the client...\n");

    func(connfd);

    close(sockfd);
}
```

Client

```
#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>
#define MAX 80
#define PORT 3000
#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')
            ;
```

```
//  input is recived and sent using the write command
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
// message from the client is read using the read command
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
//if the entered message is exit, then the process is terminated
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}

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

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

    // assign IP, PORT by cast the struct again like in the server
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

// client equivalent of the bind() command, same params
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
        != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    func(sockfd);

    close(sockfd);
}
```

Output

```
./server

Socket successfully created..
Socket successfully binded..
Server listening..
Server accepted the client...
From client: Hello
         To client : Hello
From client: Testing Level 2
         To client : Testing Level 3
Exit
From client: Exit
         To client : From client: exit
         To client : exit
Server Exit...
```

```
./client

Socket successfully created..
connected to the server..
Enter the string : Hello
From Server : Hello
Enter the string : Testing Level 2
From Server : Testing Level 3
Enter the string : Exit
From Server : Exit
Enter the string : exit
quit
From Server : exit
Client Exit...
```