Experiment-10:Implementation of Multi-user chat server using TCP

serverMultiChat.c

```
#include <sys/socket.h>
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>
#include <pthread.h>

pthread_mutex_t mutex;
int clients[20];
int n=0;

void sendtoall(char *msg,int curr){
    int i;
    pthread_mutex_lock(&mutex);
    for(i = 0; i < n; i++) {
            if(clients[i] != curr) {
                    if(send(clients[i],msg,strlen(msg),0) < 0) {
                            printf("sending failure n");
                            continue;
                    }
            }
    }
    pthread_mutex_unlock(&mutex);
}

void *recvmg(void *client_sock){
    int sock = *((int *)client_sock);
    char msg[500];
    int len;
    while((len = recv(sock,msg,500,0)) > 0) {
            msg[len] = '\0';
            sendtoall(msg,sock);
    }

}

int main(){
    struct sockaddr_in ServerIp;
    pthread_t recvt;
    int sock=0 , Client_sock=0;

    ServerIp.sin_family = AF_INET;
```

```
    ServerIp.sin_port = htons(1234);
    ServerIp.sin_addr.s_addr = inet_addr("127.0.0.1");
    sock = socket( AF_INET , SOCK_STREAM, 0 );
    if( bind( sock, (struct sockaddr *)&ServerIp, sizeof(ServerIp)) == -1 )
            printf("cannot bind, error!! n");
    else
            printf("Server Started\n");

    if( listen( sock ,20 ) == -1 )
            printf("listening failed n");

    while(1){
            if( (Client_sock = accept(sock, (struct sockaddr *)NULL,NULL)) < 0 )
                    printf("accept failed  n");
            pthread_mutex_lock(&mutex);
            clients[n]= Client_sock;
            n++;
            // creating a thread for each client
            pthread_create(&recvt,NULL,(void *)recvmg,&Client_sock);
            pthread_mutex_unlock(&mutex);
    }
    return 0;

}
```

// ExecutionStep:

Terminal-1

>> gcc serverMultiChat.c -o ser -pthread

>> ./ser

```
alwin@debian:~/Downloads$ gcc serverMultiChat_TCP.c -o ser
 -pthread
alwin@debian:~/Downloads$ ./ser
Server Started
```

clientMulitChat.c

```
#include <sys/socket.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <pthread.h>
```

```
char msg[500];

void *recvmg(void *my_sock)
{
    int sock = *((int *)my_sock);
    int len;
    // client thread always ready to receive message
    while((len = recv(sock,msg,500,0)) > 0) {
            msg[len] = '\0';
            fputs(msg,stdout);
    }
}

int main(int argc,char *argv[]){
    pthread_t recvt;
    int len;
    int sock;
    char send_msg[500];
    struct sockaddr_in ServerIp;
    char client_name[100];
    strcpy(client_name, argv[1]);
    sock = socket( AF_INET, SOCK_STREAM,0);
    ServerIp.sin_port = htons(1234);
    ServerIp.sin_family= AF_INET;
    ServerIp.sin_addr.s_addr = inet_addr("127.0.0.1");
    if( (connect( sock ,(struct sockaddr *)&ServerIp,sizeof(ServerIp))) == -1 )
            printf("n connection to socket failed n");

    //creating a client thread which is always waiting for a message
    pthread_create(&recvt,NULL,(void *)recvmg,&sock);

    //ready to read a message from console
    while(fgets(msg,500,stdin) > 0) {
            strcpy(send_msg,client_name);
            strcat(send_msg,":");
            strcat(send_msg,msg);
            len = write(sock,send_msg,strlen(send_msg));
            if(len < 0)
                    printf("n message not sent n");
    }

    //thread is closed
    pthread_join(recvt,NULL);
    close(sock);
    return 0;
}
```

// ExecutionStep:
Terminal-2
>> gcc clientMultiChat.c -o cli -pthread
>> ./cli client-1

```
alwin@debian:~/Downloads$ gcc clientMultiChat_TCP.c -o cli
 -pthread
alwin@debian:~/Downloads$ ./cli client-2
Hai I am client-2
client-1:Hai I am client-1
```

Terminal-3
>> gcc clientMultiChat.c -o cli -pthread
>> ./cli client-2

```
alwin@debian:~/Downloads$ gcc clientMultiChat_TCP.c
-o client1 -pthread
alwin@debian:~/Downloads$ ./cli client-1
client-2:Hai I am client-2
Hai I am client-1
```

Experiment-11: implementation of Echo Server using TCP

echoServer_TCP.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 4444
```

```
int main(){

    int sockfd, ret;
     struct sockaddr_in serverAddr;

    int newSocket;
    struct sockaddr_in newAddr;

    socklen_t addr_size;

    char buffer[1024];
    pid_t childpid;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0){
            printf("[-]Error in connection.\n");
            exit(1);
    }
    printf("[+]Server Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    ret = bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if(ret < 0){
            printf("[-]Error in binding.\n");
            exit(1);
    }
    printf("[+]Bind to port %d\n", 4444);

    if(listen(sockfd, 10) == 0){
            printf("[+]Listening....\n");
    }else{
            printf("[-]Error in binding.\n");
    }


    while(1){
            newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
            if(newSocket < 0){
                    exit(1);
            }
            printf("Connection accepted from %s:%d\n", inet_ntoa(newAddr.sin_addr),
ntohs(newAddr.sin_port));
```

```
        if((childpid = fork()) == 0){
                close(sockfd);

                while(1){
                        recv(newSocket, buffer, 1024, 0);
                        if(strcmp(buffer, ":exit") == 0){
                                printf("Disconnected from %s:%d\n",
inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));
                                break;
                        }else{
                                printf("Client: %s\n", buffer);
                                send(newSocket, buffer, strlen(buffer), 0);
                                bzero(buffer, sizeof(buffer));
                        }
                }
        }

  }

  close(newSocket);


  return 0;
}
```

//ExecutionSteps:

Terminal-1

>> gcc echoServer_TCP.c -o ser

>>./ser

```
alwin@debian:~/Downloads$ gcc echoServer_TCP.c -o ser
alwin@debian:~/Downloads$ ./ser
[+]Server Socket is created.
[+]Bind to port 4444
[+]Listening....
Connection accepted from 252.127.0.0:45525
Connection accepted from 127.0.0.1:42994
Client: Hai
Client: Hello
Disconnected from 127.0.0.1:42994
Disconnected from 252.127.0.0:45525
```

echoClient_TCP.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 4444

int main(){

    int clientSocket, ret;
    struct sockaddr_in serverAddr;
    char buffer[1024];

    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if(clientSocket < 0){
            printf("[-]Error in connection.\n");
            exit(1);
    }
    printf("[+]Client Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    ret = connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if(ret < 0){
            printf("[-]Error in connection.\n");
            exit(1);
    }
    printf("[+]Connected to Server.\n");

    while(1){
            printf("Client: \t");
            scanf("%s", &buffer[0]);
            send(clientSocket, buffer, strlen(buffer), 0);

            if(strcmp(buffer, ":exit") == 0){
                    close(clientSocket);
                    printf("[-]Disconnected from server.\n");
                    exit(1);
            }

            if(recv(clientSocket, buffer, 1024, 0) < 0){
                    printf("[-]Error in receiving data.\n");
            }else{
                    printf("Server: \t%s\n", buffer);
```

```
        }
    }

    return 0;
}
```

//ExecutionSteps:

Terminal-2

>> gcc echoClient_TCP.c -o cli

>>./cli client-1

```
alwin@debian:~/Downloads$ gcc echoClient_TCP.c -o cli
alwin@debian:~/Downloads$ ./cli client-1
[+]Client Socket is created.
[+]Connected to Server.
Client:         Hello
Server:         Hello
Client:         :exit
[-]Disconnected from server.
alwin@debian:~/Downloads$
```

Terminal-3

>> gcc echoServer_TCP.c -o ser

>>./cli client-2

```
alwin@debian:~/Downloads$ gcc echoClient_TCP.c -o cli
alwin@debian:~/Downloads$ ./cli client-2
[+]Client Socket is created.
[+]Connected to Server.
Client:         Hai
Server:         Hai
Client:         :exit
[-]Disconnected from server.
alwin@debian:~/Downloads$
```

Experiment-12: Implementation of Broadcast server using TCP

bserverTCP.c

```
#include<stdio.h>
#include<arpa/inet.h>
```

```
#include <unistd.h>
#include<string.h>
#define port 5000
void main()
{
   int i,n;
   printf("Enter the no of clients:");
   scanf("%d",&n);
   int sersocket,newsocket[n],s,size;
   char buffer[100];
   struct sockaddr_in serveraddr,newaddr;
   sersocket=socket(PF_INET,SOCK_STREAM,0);
   if(sersocket>0)
         printf("\n[+]server socket is created");
   serveraddr.sin_family= PF_INET;
   serveraddr.sin_port= htons(port);
   serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
   s=bind(sersocket,(struct sockaddr *)&serveraddr,sizeof(serveraddr));
   if(s==0)
         printf("\n[+]bind success");
   listen(sersocket,n);
   size=sizeof(newaddr);
   printf("\n[+]Broadcast Server is ready");
   for(i=0;i<n;i++)
         {
         newsocket[i]=accept(sersocket,(struct sockaddr *)&newaddr,&size);
         printf("\n[+]Connection accepted from %s:%d\n",
inet_ntoa(newaddr.sin_addr),ntohs(newaddr.sin_port));
         }
   while(1)
   {
   printf("\nEnter Broadcast Message:");
   scanf("%s",buffer);
   for(i=0;i<n;i++)
   {
         send(newsocket[i],buffer,sizeof(buffer),0);
   }
   if(strcmp(buffer,"bye")==0)
         break;
   }
for(i=0;i<n;i++)
   close(newsocket[i]);
}
```

//ExecutionSteps:

Terminal-1

>> gcc bserverTCP.c -o ser

>>./ser

```
alwin@debian:~/Downloads$ gcc bserverTCP.c -o ser
alwin@debian:~/Downloads$ ./ser
Enter the no of clients:2

[+]server socket is created
[+]bind success
[+]Broadcast Server is ready
[+]Connection accepted from 127.0.0.1:59884

[+]Connection accepted from 127.0.0.1:37960

Enter Broadcast Message:Hai

Enter Broadcast Message:Hello

Enter Broadcast Message:bye
alwin@debian:~/Downloads$
```

bclientTCP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include <unistd.h>
#include<string.h>
#define port 5000
void main()
{
   struct sockaddr_in serveraddr;
   int clisocket;
   char buffer[100];
   clisocket=socket(PF_INET,SOCK_STREAM,0);
         printf("\n[+]client socket created");
   serveraddr.sin_family= PF_INET;
   serveraddr.sin_port= htons(port);
   serveraddr.sin_addr.s_addr=inet_addr("127.0.0.1");
   connect(clisocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
   printf("\n[+]Connected to Broadcast Server.");
   while(1)
   {
   recv(clisocket,buffer,1024,0);
   printf("\ndata received is: %s\n",buffer);
   if(strcmp(buffer,"bye")==0)
         break;
   }
   close(clisocket);
}
```

//ExecutionSteps:

Terminal-2

>> gcc bClientTCP.c -o cli

>>./cli

```
alwin@debian:~/Downloads$ gcc bclientTCP.c -o cli
alwin@debian:~/Downloads$ ./cli

[+]client socket created
[+]Connected to Broadcast Server.
data received is: Hai

data received is: Hello

data received is: bye
alwin@debian:~/Downloads$
```

Terminal-3

>> gcc bclientTCP.c -o cli

>>./cli

```
alwin@debian:~/Downloads$ gcc bclientTCP.c -o cli
alwin@debian:~/Downloads$ ./cli

[+]client socket created
[+]Connected to Broadcast Server.
data received is: Hai

data received is: Hello

data received is: bye
alwin@debian:~/Downloads$
```