Lab09

Question01:

Server.c

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h> //socket
#include <arpa/inet.h> //inet_addr
int current_clients=0;
int total_clients=3;
void* echo_back (void *arg)
{                    int flag=1;
                     current_clients++;
                     char server_message[2000], client_message[2000];

     memset(server_message,'\0',sizeof(server_message));
     memset(client_message,'\0',sizeof(client_message));     // Set all bits of the padding field//
     int *client_sock_ptr = (int *)arg;
          int client_sock = *client_sock_ptr;
           if (client_sock < 0)
                     {
                     printf("Accept Failed. Error!!!!!!\n");
                     return;
                     }

     while(flag)
     {
     if (recv(client_sock, client_message, sizeof(client_message),0) < 0)
```

```
        {
                printf("Receive Failed. Error!!!!!\n");
                return -1;
        }
        printf("client number : %d\n",current_clients);
        printf("Client Message: %s\n",client_message);
        if(strcmp(client_message,"disconnect")==0)
        {
        strcpy(server_message, "disconnected\n");
        flag=0;
        }
        else
        {
        strcpy(server_message, client_message);
        }
        if (send(client_sock, server_message, strlen(client_message),0)<0)
        {
                printf("Send Failed. Error!!!!!\n");
                return -1;
        }

        memset(server_message,'\0',sizeof(server_message));
        memset(client_message,'\0',sizeof(client_message));
        }
         close(client_sock);
    current_clients--;
    pthread_exit(NULL);

}
int main(void)
{
```

```
int socket_desc, client_sock, client_size;
struct sockaddr_in server_addr, client_addr;        //SERVER ADDR will have all the server address

        pthread_t thread[total_clients];


//Creating Socket

socket_desc = socket(AF_INET, SOCK_STREAM, 0);

if(socket_desc < 0)
{
    printf("Could Not Create Socket. Error!!!!!\n");
    return -1;
}

printf("Socket Created\n");

//Binding IP and Port to socket

server_addr.sin_family = AF_INET;            /* Address family = Internet */
server_addr.sin_port = htons(2000);            // Set port number, using htons function to use
proper byte order */
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");    /* Set IP address to localhost */


                // BINDING FUNCTION

if(bind(socket_desc, (struct sockaddr*)&server_addr, sizeof(server_addr))<0)    // Bind the
address struct to the socket.  /
```

```
                                //bind() passes file descriptor, the address structure,and the length of the
address structure
    {
        printf("Bind Failed. Error!!!!!\n");
        return -1;
    }

    printf("Bind Done\n");

    //Put the socket into Listening State
    while(1){
    if(listen(socket_desc, 1) < 0)                    //This listen() call tells the socket to listen to the
incoming connections.
   // The listen() function places all incoming connection into a "backlog queue" until accept() call
accepts the connection.
    {
        printf("Listening Failed. Error!!!!!\n");
        return -1;
    }

    printf("Listening for Incoming Connections.....\n");

    //Accept the incoming Connections

    client_size = sizeof(client_addr);



    client_sock = accept(socket_desc, (struct sockaddr*)&client_addr, &client_size);        // heree
particular client k liye new socket create kr rhaa ha

    if (client_sock < 0)
```

```
        {
            printf("Accept Failed. Error!!!!!!\n");
            return -1;
        }
        if(current_clients<total_clients)
        {
            pthread_create(&thread[current_clients],NULL,echo_back,(void*)&client_sock);
        }
        }
        close(socket_desc);
int i;
    for (i = 0; i < total_clients; i++)
        pthread_join(thread[i], NULL);

        return 0;
}
```

```
ifra@ifra-virtual-machine:~/cn$ ./s
Socket Created
Bind Done
Listening for Incoming Connections.....
Listening for Incoming Connections.....
client number : 1
Client Message: hi there
client number : 1
Client Message: hello from the client
client number : 1
Client Message: disconnect
```

Question01 (client.c):

```
#include <stdio.h>

#include <string.h>

#include <sys/socket.h> //socket

#include <arpa/inet.h> //inet_addr


int main(void)

{                       int j=1;

        int socket_desc;

        struct sockaddr_in server_addr;

        char server_message[2000], client_message[2000];


        //Cleaning the Buffers


        memset(server_message,'\0',sizeof(server_message));

        memset(client_message,'\0',sizeof(client_message));


        //Creating Socket


        socket_desc = socket(AF_INET, SOCK_STREAM, 0);


        if(socket_desc < 0)

        {

                printf("Could Not Create Socket. Error!!!!!\n");

                return -1;

        }


        printf("Socket Created\n");


        //Specifying the IP and Port of the server to connect
```

```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(2000);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

//Now connecting to the server accept() using connect() from client side

if(connect(socket_desc, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
{
	printf("Connection Failed. Error!!!!!");
	return -1;
}

printf("Connected\n");

//Get Input from the User
while(j){
printf("Enter Message: ");
gets(client_message);				//One is that gets() will only get character string data.
					//		will get only one variable at a time.

				//  reads characters from stdin and loads them into str
//Send the message to Server

if(send(socket_desc, client_message, strlen(client_message),0) < 0)
{
	printf("Send Failed. Error!!!!\n");
	return -1;
}
if(strcmp(client_message,"disconnect")==0)
{
j=0;
```

```
    strcpy(server_message,"disconnected\n");
    }
    else{strcpy(server_message,client_message);}
    //Receive the message back from the server

    if(recv(socket_desc, server_message, sizeof(server_message),0) < 0)
    {
        printf("Receive Failed. Error!!!!!\n");
        return -1;
    }

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

    memset(server_message,'\0',sizeof(server_message));
    memset(client_message,'\0',sizeof(client_message));
    }
    //Closing the Socket

    close(socket_desc);

    return 0;
}
```

```
ifra@ifra-virtual-machine:~/cn$ ./c
Socket Created
Connected
Enter Message: hi there
Server Message: hi there
Enter Message: hello from the client
Server Message: hello from the client
Enter Message: disconnect
Server Message: disconnected

ifra@ifra-virtual-machine:~/cn$
```

Question02(server.c):

```
#include <stdio.h>

#include <string.h>

#include <sys/socket.h> //socket

#include <arpa/inet.h> //inet_addr

#include <pthread.h>


void *cast_vote(void *arg)

{

   int client_sock = (int)arg;

   char server_message[2000], client_message[2000];


   //Cleaning the Buffers

   memset(server_message, '\0', sizeof(server_message));

   memset(client_message, '\0', sizeof(client_message));


   //Receive Name/CNIC from the client

   if (recv(client_sock, client_message, sizeof(client_message), 0) < 0)

   {

      printf("Receive Failed. Error!!!!!\n");

   }


      //tokenizing the client_message

            char delimiter='/';

            char*p[3];

      p[0]=strtok(client_message,&delimiter);

      int z=0;

      while(p[z]!=NULL)

      {

      z++;

      p[z]=strtok(NULL,&delimiter);
```

```
}
p[z]='\0';

FILE*v=fopen("./Voters_List.txt","r");
if(v==NULL)
{
printf("file not opened");
}
char buffer[255];
int flag=0;
char *q[3];
int a=0;
printf("fine ");
    while(fgets(buffer, sizeof(buffer), v)!=NULL)
    {
    q[0]=strtok(buffer,&delimiter);
    while(q[a]!=NULL)
    {
    a++;
    q[a]=strtok(NULL,&delimiter);
    }
    q[a]='\0';
    if(strcmp(p[0],q[0])==0 && strcmp(p[1],q[1])==0)
    {

    flag=1;
    break;
    }
    }

    fclose(v);
```

```
    if(flag==1)//voter is present in voters_list
    {



        memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

strcpy(server_message, "Welcome Voter.\n");

if (send(client_sock, server_message, strlen(server_message), 0) < 0)
{
   printf("Send Failed. Error!!!!!\n");
}
//checking if voter has already casted vote or not
char buff2[255];
char*r[3];
int already_casted=0;
FILE*o=fopen("./out.txt","r");
int b=0;
while(fgets(buff2,sizeof(buff2),o)!=NULL)
{
r[0]=strtok(buff2,',');

while(r[b]!=NULL)
    {
    b++;
    r[b]=strtok(NULL,',');
    }
    r[b]='\0';
    if(strcmp(p[0],r[0])==0)
    {
```

```
  already_casted=1;
  break;
  }
  }
  fclose(o);
  if(already_casted==0)
  {
  char display_candidates[100];
  char buff3[255];
  FILE*c=fopen("./Candidates_List.txt","r");
  strcpy(display_candidates,"the candidates are: \n");
  while(fgets(buff3,sizeof(buff3),c)!=NULL)
  {
  strcat(display_candidates,buff3);
  }
  fclose(c);
            memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

strcpy(server_message, display_candidates);

if (send(client_sock, server_message, strlen(server_message), 0) < 0)
{
   printf("Send Failed. Error!!!!!\n");
}

//Cleaning the Buffers
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

//Receive candidate's symbol from client
```

```
if (recv(client_sock, client_message, sizeof(client_message), 0) < 0)
{
   printf("Receive Failed. Error!!!!!\n");
}
//updating output.txt
char vote[10];
char voter_symbol[100];
strcpy(vote,client_message);
strcpy(voter_symbol,q[0]);
strcat(voter_symbol,',');
strcat(voter_symbol,vote);
strcat(voter_symbol,"\n");
o=fopen("./out.txt","a");
fputs(voter_symbol,o);
fclose(o);
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

strcpy(server_message, "vote casted.\n");

if (send(client_sock, server_message, strlen(server_message), 0) < 0)
{
   printf("Send Failed. Error!!!!!\n");
}

  }
  else
  {
     memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));
```

```
            strcpy(server_message,"You have already casted a vote.\n");

            if (send(client_sock, server_message, strlen(server_message), 0) < 0)
            {
                printf("Send Failed. Error!!!!!\n");
            }

              }
        }
        else
        {
        memset(server_message, '\0', sizeof(server_message));
        memset(client_message, '\0', sizeof(client_message));

        strcpy(server_message, "you donot belong to the voters\n");

        if (send(client_sock, server_message, strlen(server_message), 0) < 0)
        {
            printf("Send Failed. Error!!!!!\n");
        }

        }

    close(client_sock);
    pthread_exit(NULL);
}

int main(void)
{
    int socket_desc, client_sock, client_size;
    struct sockaddr_in server_addr, client_addr;
```

```
char server_message[2000], client_message[2000];
pthread_t thread;

//Cleaning the Buffers
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

//Creating Socket
socket_desc = socket(AF_INET, SOCK_STREAM, 0);

if (socket_desc < 0)
{
   printf("Could Not Create Socket. Error!!!!!\n");
   return -1;
}

printf("Socket Created\n");

//Binding IP and Port to socket
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(2000);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

if (bind(socket_desc, (struct sockaddr *)&server_addr,sizeof(server_addr)) < 0)
{
   printf("Bind Failed. Error!!!!!\n");
   return -1;
}

printf("Bind Done\n");
```

```
    //Put the socket into Listening State
    if (listen(socket_desc, 1) < 0)
    {
        printf("Listening Failed. Error!!!!!\n");
        return -1;
    }

    printf("Listening for Incoming Connections.....\n");

    while (1)
    {
        //Accept the incoming Connections
        client_size = sizeof(client_addr);
        client_sock = accept(socket_desc, (struct sockaddr*)&client_addr,&client_size);

        if (client_sock < 0)
        {
            printf("Accept Failed. Error!!!!!!\n");
            return -1;
        }

        //printf("Client Connected with IP: %s and Port No: %i\n", inet_ntoa(client_addr.sin_addr);
ntohs(client_addr.sin_port);

        strcpy(client_message, "Connection Established!");

        // Send the connection message back to client
        if (send(client_sock, client_message, strlen(client_message), 0) < 0)
        {
            printf("Send Failed. Error!!!!!\n");
            return -1;
```

```
        }

        //Cleaning the Buffers
        memset(server_message, '\0', sizeof(server_message));
        memset(client_message, '\0', sizeof(client_message));

        //create new thread for each client
        pthread_create(&thread, NULL, cast_vote, (void *)client_sock);
    }
    pthread_join(thread, NULL); //Suspends main thread until termination of threads

    //Cleaning the Buffers
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    //Closing the Socket
    //close(socket_desc);
    return 0;
}
```

Question02(client.c):

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

int main(void)
{
    int socket_desc;
    struct sockaddr_in server_addr;
```

```
char server_message[2000], client_message[2000];

//Cleaning the Buffers
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

//Creating Socket
socket_desc = socket(AF_INET, SOCK_STREAM, 0);

if (socket_desc < 0)
{
   printf("Could Not Create Socket. Error!!!!!\n");
   return -1;
}

printf("Socket Created\n");

//Specifying the IP and Port of the server to connect
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(2000);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

//Now connecting to the server accept() using connect() from client side

if (connect(socket_desc, (struct sockaddr *)&server_addr,sizeof(server_addr)) < 0)
{
   printf("Connection Failed. Error!!!!!");
   return -1;
}

//Receive the connection message back from the server
```

```
if (recv(socket_desc, server_message, sizeof(server_message), 0) < 0)
{
   printf("Receive Failed. Error!!!!!\n");
   return -1;
}

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

//Cleaning the Buffers
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

//Get Input from the User
printf("Enter Message: ");
fgets(client_message,sizeof(client_message),stdin);

//Send credentials back to Server
if (send(socket_desc, client_message, strlen(client_message), 0) < 0)
{
   printf("Send Failed. Error!!!!\n");
   return -1;
}

//Cleaning the Buffers
memset(server_message, '\0', sizeof(server_message));
memset(client_message, '\0', sizeof(client_message));

//Receive the message back from the server
if (recv(socket_desc, server_message, sizeof(server_message), 0) < 0)
{
   printf("Receive Failed. Error!!!!!\n");
```

```
    return -1;
}

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

if (strcmp(server_message, "you donot belong to the voters.\n") != 0)
{
    //Receive the message back from the server
    if (recv(socket_desc, server_message, sizeof(server_message), 0) < 0)
    {
        printf("Receive Failed. Error!!!!!\n");
        return -1;
    }

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

    if (strcmp(server_message, "You have already casted avote.\n") != 0)
    {
        //Cleaning the Buffers
        memset(server_message, '\0', sizeof(server_message));
        memset(client_message, '\0', sizeof(client_message));

        //Get Input Symbol from the User
        printf("Enter the candidate's symbol: ");
        fgets(client_message,sizeof(client_message),stdin);

        //Send the symbol back to server
        if (send(socket_desc, client_message,strlen(client_message), 0) < 0)
        {
            printf("Send Failed. Error!!!!\n");
            return -1;
```

```
        }

        //Receive confirmation from server
        if (recv(socket_desc, server_message,sizeof(server_message), 0) < 0)
        {
            printf("Receive Failed. Error!!!!!\n");
            return -1;
        }

        printf("Server Message: %s\n\n", server_message);
      }
    }

    return 0;
}
```