****

**NETWORKS LAB**

**EXERCISE 3**

Name: Jayannthan P T

Dept: CSE ‘A’

Roll No.: 205001049

Multiuser Chat Using UDP

**Aim:**

Develop a chat application between a client and server using UDP. Update the program to support multiple clients (using fd\_set() and select() functions of C.)

**Algorithm:**

**SERVER**

1. Create a UDP socket using socket() system call.

2. Bind() is used to bind the socket with a specified address defined by sockaddr\_in pointer, with the address, family, port set accordingly, bzero() is used to clear the address pointer initially.

3. Initialize a descriptor set for select and calculate a maximum of 3 descriptor for which server will wait

4. Call select and get the ready descriptor (UDP)

5. Handle new connection and receive the data gram

**CLIENT**

1. Create a UDP socket using socket() system call.

2. Send message to server

3. Wait until response from server is received.

4. Close socket descriptor and exit.

**Code:**

**Server**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <netdb.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <fcntl.h>

**int** main(**int** argc, **char** **const** \*argv**[]**)

{

**int** port = atoi(argv[1]);

**int** sockfd, csize, arr[30], pre = 0, flag, p, i;

**char** buffer[1024];

**struct** sockaddr\_in serveraddr, clientaddr;

    fd\_set readfds;

    for (**int** i = 0; i < 30; i++)

        arr[i] = 0;

    sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (sockfd == -1)

    {

        printf("SOCKET creation failed !!\n");

        exit(0);

    }

    else

        printf("socket creation succesful\n");

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(port);

    if ((bind(sockfd, (**struct** sockaddr \*)&serveraddr, sizeof(serveraddr))) != 0)

    {

        printf("socket binding failed\n");

        exit(0);

    }

    else

    {

        printf("socket binding successful\n");

    }

    bzero(&serveraddr, sizeof(**struct** sockaddr\_in));

    FD\_ZERO(&readfds);

    while (1)

    {

        FD\_SET(sockfd, &readfds);

        csize = sizeof(clientaddr);

        select(sockfd + 1, &readfds, NULL, NULL, NULL);

        recvfrom(sockfd, buffer, 1024, 0, (**struct** sockaddr \*)&clientaddr, &csize);

        p = ntohs(clientaddr.sin\_port);

        flag = 0;

        for (i = 0; i < 30; i++)

            if (arr[i] == p)

            {

                p = i + 1;

                flag = 1;

                break;

            }

        if (flag == 0)

        {

            arr[pre] = p;

            pre++;

            p = pre;

        }

        if (strcmp(buffer, "exit") == 0)

        {

            printf("\n Client %d closed the chat\n", p);

        }

        else

        {

            printf("\nMessage from Client %d: %s\n", p, buffer);

            csize = sizeof(serveraddr);

**char** strData[255];

            printf("\nEnter reply:");

            scanf(" %[^\n]s", strData);

            sendto(sockfd, strData, sizeof(strData), 0, (**struct** sockaddr \*)&clientaddr, csize);

*// printf("Message sent: %s\n", strData);*

        }

    }

    close(sockfd);

    return 0;

}

**Client**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <netdb.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <fcntl.h>

**int** main(**int** argc, **char** **const** \*argv**[]**)

{

**int** port = atoi(argv[1]);

**int** sockfd, ssize;

**char** buffer[1024];

**struct** sockaddr\_in serveraddr;

    sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (sockfd == -1)

    {

        printf("SOCKET creation failed !!\n");

        exit(0);

    }

    else

        printf("socket creation succesful\n");

    serveraddr.sin\_family = AF\_INET;

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

    serveraddr.sin\_port = htons(port);

**int** connected = 1;

**char** strData[255];

    while (connected)

    {

        ssize = sizeof(serveraddr);

        printf("\nEnter data to send to server:");

        scanf(" %[^\n]s", buffer);

        sendto(sockfd, buffer, sizeof(buffer), 0, (**struct** sockaddr \*)&serveraddr, ssize);

        recvfrom(sockfd, strData, sizeof(strData), 0, (**struct** sockaddr \*)&serveraddr, &ssize);

        if (strcmp(buffer, "exit") == 0)

        {

            connected = 0;

        }

        else

        {

            printf("\nMessage from server: %s\n", strData);

        }

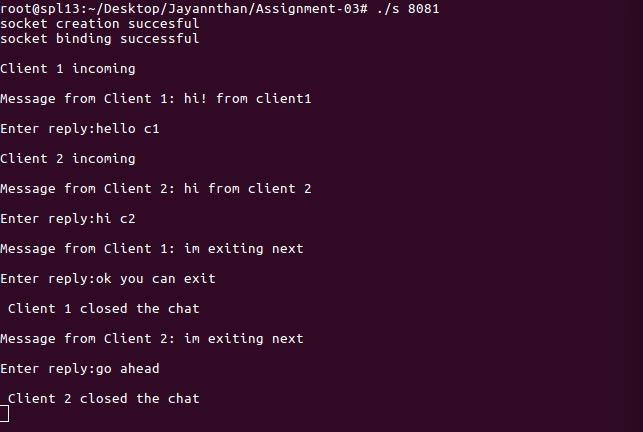
    }

    close(sockfd);

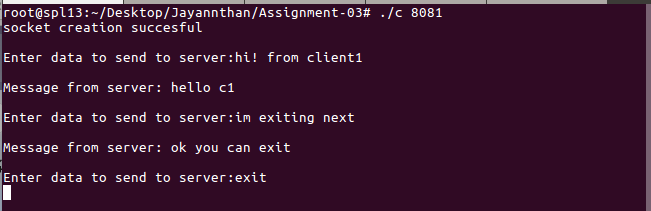
}

**Output:**

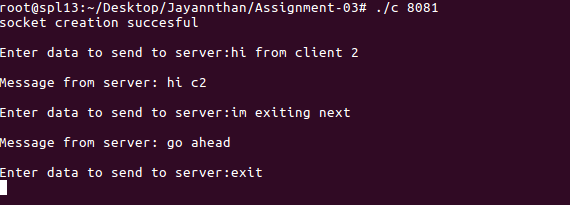
Server :



Client 1:



Client 2:



**Learning outcome:**

Learnt to create UDP connection using sockets

Learnt to communicate between server and multiple clients using socket