SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

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

```c
#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>
```

```c
#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 :

```
root@spl13:~/Desktop/Jayannthan/Assignment-03# ./s 8081
socket creation succesful
socket binding successful

Client 1 incoming

Message from Client 1: hi! from client1

Enter reply:hello c1

Client 2 incoming

Message from Client 2: hi from client 2

Enter reply:hi c2

Message from Client 1: im exiting next

Enter reply:ok you can exit

 Client 1 closed the chat

Message from Client 2: im exiting next

Enter reply:go ahead

 Client 2 closed the chat
```

Client 1:

```
root@spl13:~/Desktop/Jayannthan/Assignment-03# ./c 8081
socket creation succesful

Enter data to send to server:hi! from client1

Message from server: hello c1

Enter data to send to server:im exiting next

Message from server: ok you can exit

Enter data to send to server:exit
```

Client 2:

```
root@spl13:~/Desktop/Jayannthan/Assignment-03# ./c 8081
socket creation succesful

Enter data to send to server:hi from client 2

Message from server: hi c2

Enter data to send to server:im exiting next

Message from server: go ahead

Enter data to send to server:exit
```

Learning outcome:

Learnt to create UDP connection using sockets

Learnt to communicate between server and multiple clients using socket