Submitted by – Paras Jain (2018KUCP1006)

Computer Networks Lab

Assignment -2 (Socket Programming)

Server Side Code:

```
//Submitted By Paras Jain (2018KUCP1006)
//Server side code
#include <stdio.h>
// It is standard input and output library
#include <stdlib.h>
// It includes functions regarding memory allocation
#include <string.h>
// It contains string functions
#include <errno.h>
//It defines macros for reporting and retrieving error conditions through error codes
#include <time.h>
// It contains various functions for manipulating date and time
#include <unistd.h>
//It contains various constants
#include <sys/types.h>
//It contains a number of basic derived types that should be used whenever appropriate
#include <arpa/inet.h>
// It defines in_addr structure
#include <sys/socket.h>
// It is for socket creation
#include <netinet/in.h>
//It contains constants and structures needed for internet domain addresses

int main(){
    char dataSending[1025];
    // This is basically declaring data variable.
    //This is called packet in Network Communication, which contain data and send through.
    memset(dataSending, '0', sizeof(dataSending));
    //Initialising the dataSending array to char '0'
```

```
    int clintConnt = 0;
    // Client connection for file descripter of the accepted socket
    //creating socket
    //socket() function creates a new socket inside kernel and returns an integer which used as socket descriptor.
    int clintListn = socket(AF_INET, SOCK_STREAM, 0);
    // AF_INET is for IPV4 address, SOCK_STREAM is for connection-
oriented,it confirms that we are using TCP as a protocol in this communication, which is reliable communication, 0 telling kernel to
use default protocol

    struct sockaddr_in ipOfServer;
    //Creating a variable of datatype struct sockaddr_in to Provide IP address of server
    memset(&ipOfServer, '0', sizeof(ipOfServer));
    //Initialising all the member variables of structure ipOfServer with '0'
    ipOfServer.sin_family = AF_INET;
    //same as socket call (AF_INET) i.e for IPV4 address.
    ipOfServer.sin_addr.s_addr = htonl(INADDR_ANY); //bind to any local address
    ipOfServer.sin_port = htons(2017);
    // specify port to listen on, this is basically the port number of running server
//  Network'order'is'big-endian'
//  Host'order'can'be'big-'or'little-endian'
//  conversion is done using htons(), htonl():'host'to'network'short/long'
    bind(clintListn, (struct sockaddr *)&ipOfServer, sizeof(ipOfServer));
    //to bind the created socket to structure ipOfServer, we are calling bind() function,
    //which includes port, ip addresses as arguments.
    listen(clintListn, 20);
    //Telling willingness to accept connections,
    //the 2nd argument 20 says that maximum 20 number of clients can connect to that server.
    //So maximum 20 queue process can be handled by server.
    time_t clock; //  for storing time in seconds. It is a integral value.
    while(1){
        //Now the server has started.
        //Next server waits for client request by accept() function.
        clintConnt = accept(clintListn, (struct sockaddr *)NULL, NULL);
        //Whenever successfully complets (a client hits), accept()
        //returns the non-negative file descriptor of the accepted socket
        printf("\n\nHi,Iam running server.Some Client hit me\n");
```

```
        // whenever a request from client came. It will be processed here.
        clock = time(NULL);
        //This is basically used to get the current time i.e the time at which a clint hit the server.
        //This function returns the time since 00:00:00 UTC, January 1, 1970 in seconds
        //snprintf() function formats and stores a series of characters and values in the array buffer
        snprintf(dataSending, sizeof(dataSending), "%.24s\r\n", ctime(&clock));
        // It is storing the time data in the character array datasending
        //The write() function is used to write object or record (sequence of bytes) to the file.
        //A record may be an array, structure or class.
        write(clintConnt, dataSending, strlen(dataSending));
        //It is writting the data from character array datasending in the file whose descripter was
        //returned by the accept() function which was basically sent by the client
        close(clintConnt);  // closing the above file.
        sleep(1);
        // accept() runs infinite loop to keep server running always.
        //But it may eat up all CPU processing, to avoid that we have written sleep(1),
        //which server went to sleep for 1 sec.
    }
    return 0;
}
```

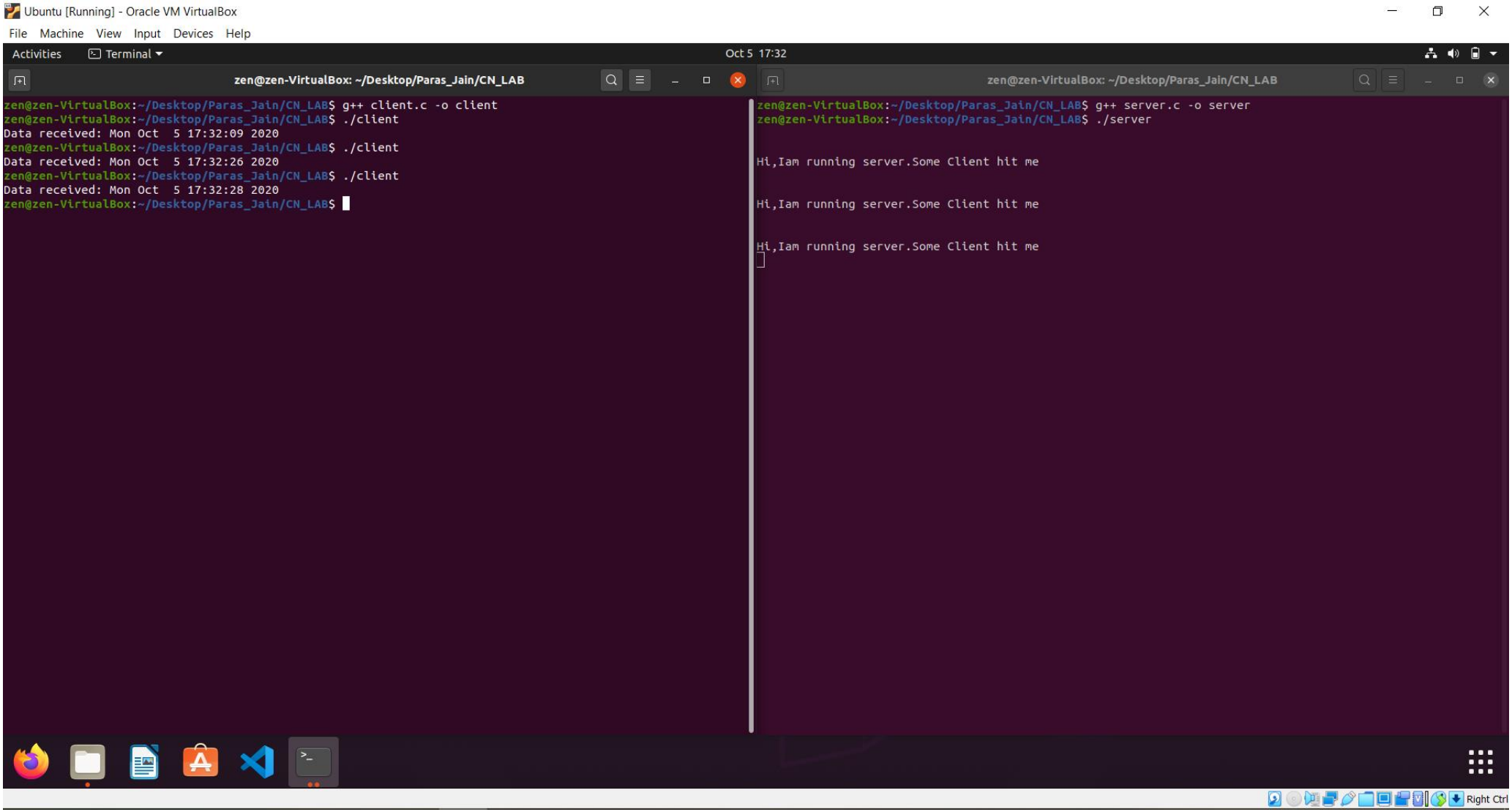
Client Side Code:

```
//Submitted By Paras Jain (2018KUCP1006)
//client side code
#include <stdio.h>
// It is standard input and output library
#include <stdlib.h>
// It includes functions regarding memory allocation
#include <string.h>
// It contains string functions
#include <errno.h>
//It defines macros for reporting and retrieving error conditions through error codes
#include <unistd.h>
//It contains various constants
#include <sys/types.h>
//It contains a number of basic derived types that should be used whenever appropriate
#include <arpa/inet.h>
// It defines in_addr structure
#include <sys/socket.h>
// It is for socket creation
#include <netinet/in.h>
//It contains constants and structures needed for internet domain addresses

int main(){
    char dataReceived[1024];
    // This is basically declaring data variable. This is called packet in Network Communication, which contain data and send through
.
    memset(dataReceived, '\0' ,sizeof(dataReceived));
    //Initialising the character array with '\0' i.e null character;
    int CreateSocket = socket(AF_INET, SOCK_STREAM, 0);
    //Since this communication is through socket, here also, we created socket.
    // AF_INET is for IPV4 address, SOCK_STREAM is for connection-
oriented,it confirms that we are using TCP as a protocol in this communication, which is reliable communication, 0 telling kernel to
use default protocol
    if(CreateSocket < 0){  // Socket function returns -ve value if it is not created
        printf("Socket not created \n");
```

```
        return 1;
    }
    struct sockaddr_in ipOfServer;
    ipOfServer.sin_family = AF_INET; //(AF_INET) i.e for IPV4 address.
    ipOfServer.sin_port = htons(2017); //  This is the port to connect with.
    ipOfServer.sin_addr.s_addr = inet_addr("127.0.0.1");
    // loopback address return address of server
    if(connect(CreateSocket, (struct sockaddr *)&ipOfServer, sizeof(ipOfServer))<0){
    //connecting to server, -ve value implies unsuccessful
        printf("Connection failed due to port and ip problems\n");
        return 1;
    }
    recv(CreateSocket, dataReceived, sizeof(dataReceived),0);   //Recieving data from the server
    printf("Data received: %s", dataReceived);  //Printing recieved data
    return 0;
}
```

Output:


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 5 17:32
zen@zen-VirtualBox: ~/Desktop/Paras_Jain/CN_LAB
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ g++ client.c -o client
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ ./client
Data received: Mon Oct 5 17:32:09 2020
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ ./client
Data received: Mon Oct 5 17:32:26 2020
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ ./client
Data received: Mon Oct 5 17:32:28 2020
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$
zen@zen-VirtualBox: ~/Desktop/Paras_Jain/CN_LAB
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ g++ server.c -o server
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB$ ./server
Hi,Iam running server.Some Client hit me
Hi,Iam running server.Some Client hit me
Hi,Iam running server.Some Client hit me
Right Ctrl