# DCCN LAB ASSIGNMENT 02

# MUAAZ SHOAIB

# FA20-BCS-074

# DATE: 04-Nov-2022

## Q1: CLIENT SERVER CHAT APPLICATION

### CODE IN CLIENT.C

#include <arpa/inet.h> // inet\_addr()

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <strings.h> // bzero()

#include <sys/socket.h>

#include <unistd.h> // read(), write(), close()

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));

bzero(buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

break;

}

}

}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// socket create and verification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr))

!= 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

// function for chat

func(sockfd);

// close the socket

close(sockfd);

}

### CODE IN SERVER.C

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int connfd)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;) {

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(connfd, buff, sizeof(buff));

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

bzero(buff, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n')

;

// and send that buffer to client

write(connfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

// Driver function

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

// socket create and verification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server accept failed...\n");

exit(0);

}

else

printf("server accept the client...\n");

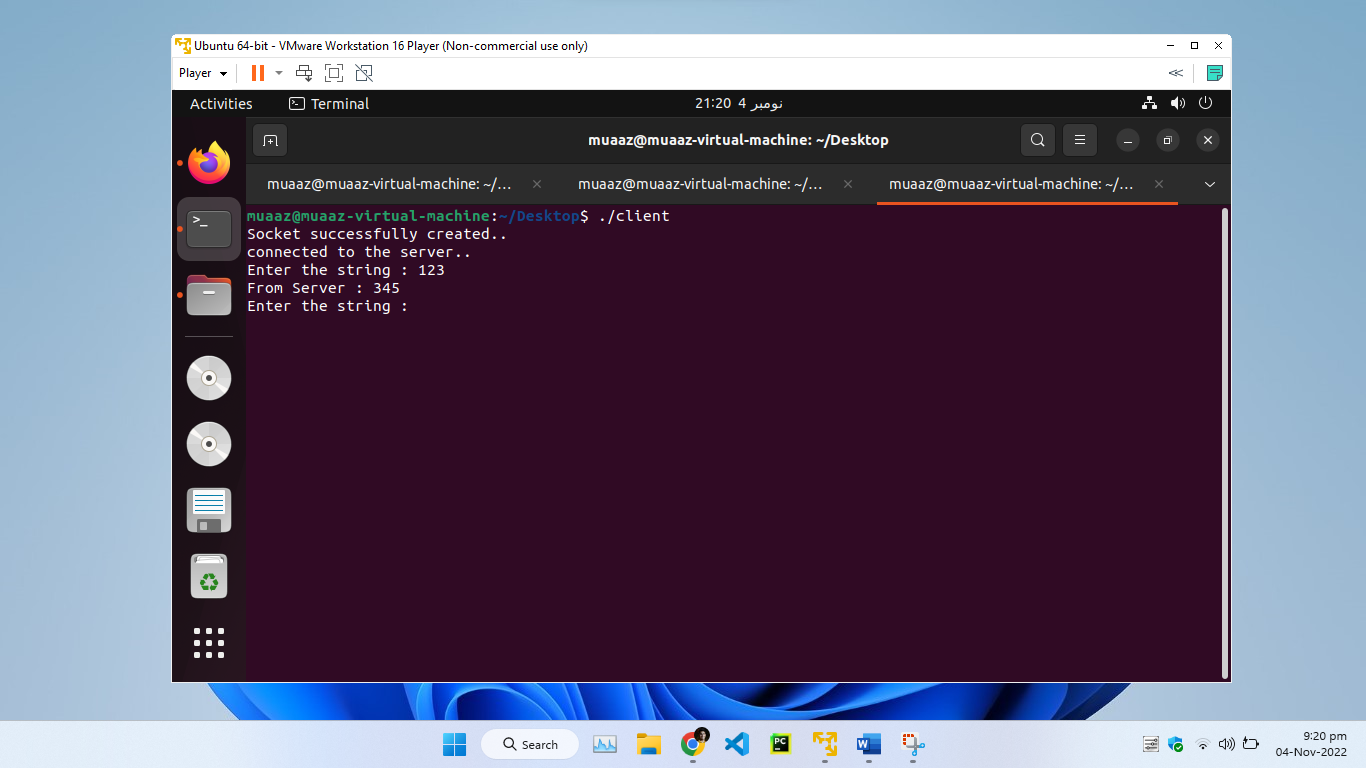
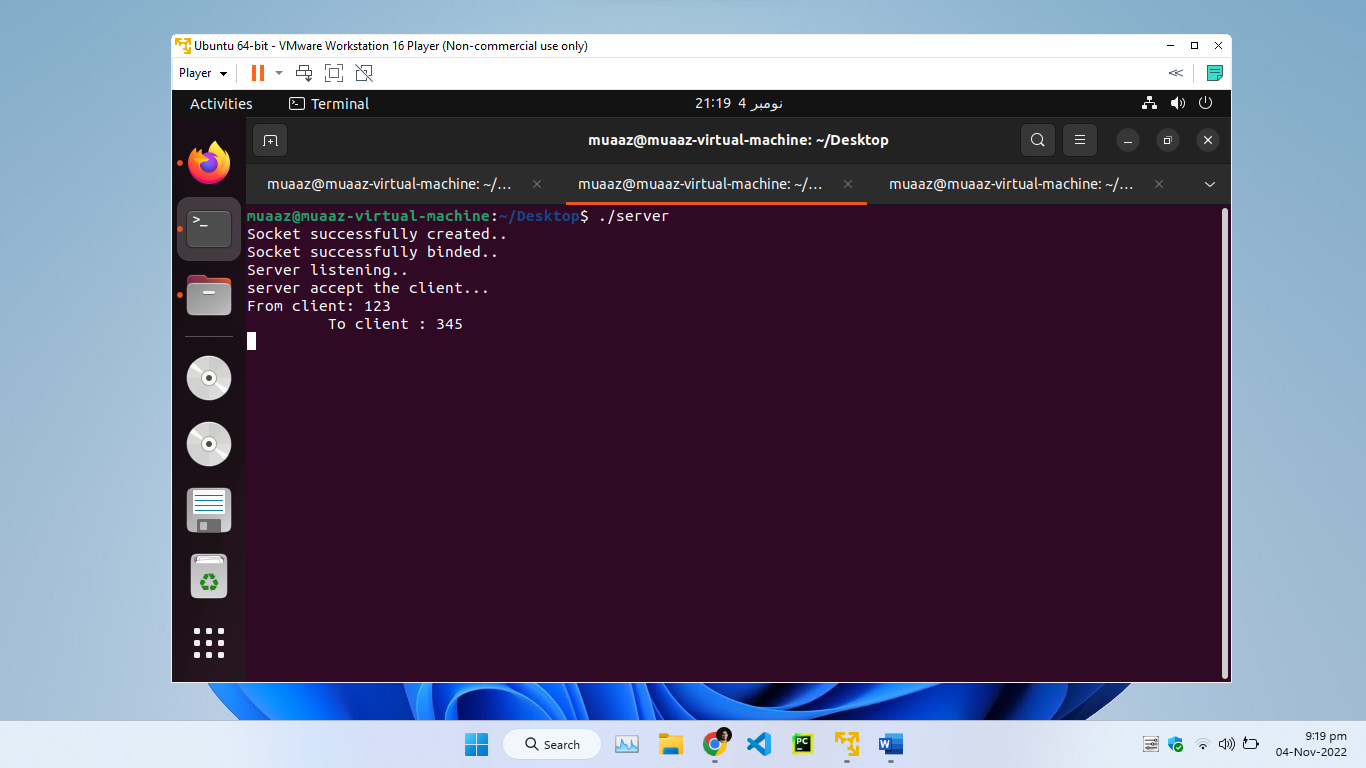
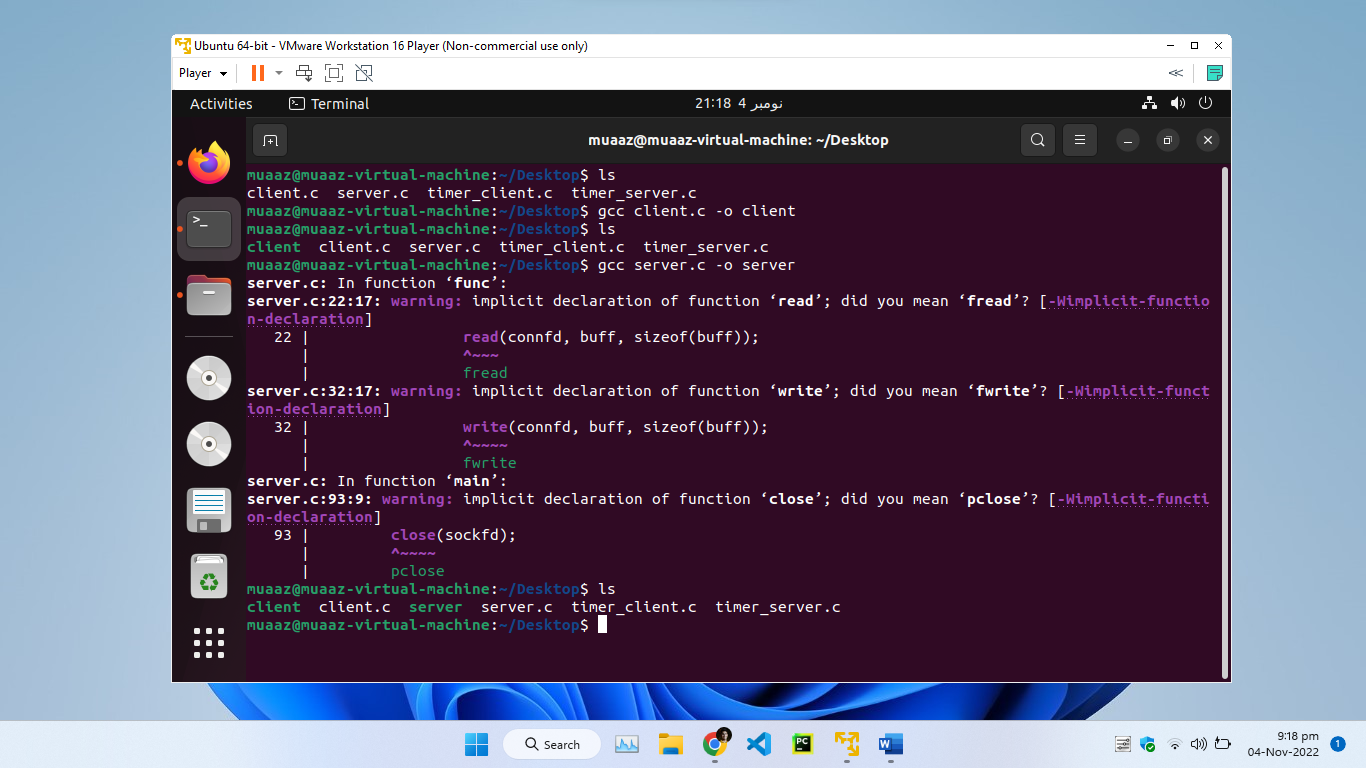
// Function for chatting between client and server

func(connfd);

// After chatting close the socket

close(sockfd);

}



## Q2: TIME SERVER APPLICATION

### CODE FOR CLIENT TIMER

#include<netinet/in.h>

#include<sys/socket.h>

main()

{

struct sockaddr\_in sa,cli;

int n,sockfd;

int len;

char buff[100];

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

if(sockfd<0)

{

printf(“Error in Socket”);

exit(0);

}

else

printf(“Socket is Opened”);

bzero(&sa,sizeof(sa));

sa.sin\_family=AF\_INET;

sa.sin\_port=htons(5600);

if(connect(sockfd,(struct sockaddr\*)&sa,sizeof(sa))<0)

{

printf(“Error in connection failed”);

exit(0);

}

else

printf(“connected successfully”):

if(n=read(sockfd,buff,sizeof(buff))<0)

{

printf(“Error in Reading”);

exit(0);

}

else

{

printf(“Message Read %s”,buff);

buff[n]=’\0’;

printf(“%s”,buff);

}

}

### CODE FOR SERVER TIMER

#include<netinet/in.h>

#include<sys/socket.h>

main( )

{

struct sockaddr\_in sa;

struct sockaddr\_in cli;

int sockfd,coontfd;

int len,ch;

char str[100];

time\_t tick;

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

if(socket<0)

{

printf(“error in socket\n”);

exit(0);

}

else

printf(“Socket Opened”);

bzero(7sa,sizeof(sa));

sa.sin\_port=htons(5600);

sa.sin\_addr.s\_addr=htonl(0);

if(bind(sockfd,(struct sockaddr\*)&sa,sizeof(sa))<0)

{

printf(“Error in binding\n”);

}

else

printf(“Binded Successfully”);

listen(sockfd,50)

for(;;)

{

len=sizeof(ch);

conntfd=accept(sockfd,(struct sockaddr\*)&cli,&len);

printf(“Accepted”);

tick=ctime(NULL);

snprintf(str,sizeof(str),”%s”,ctime(&tick));

write(conntfd,str,100);

}

}