Lab1 – Computer Networks Lab

Name: Rhea Adhikari
Reg No: 190905156
Roll No: 23

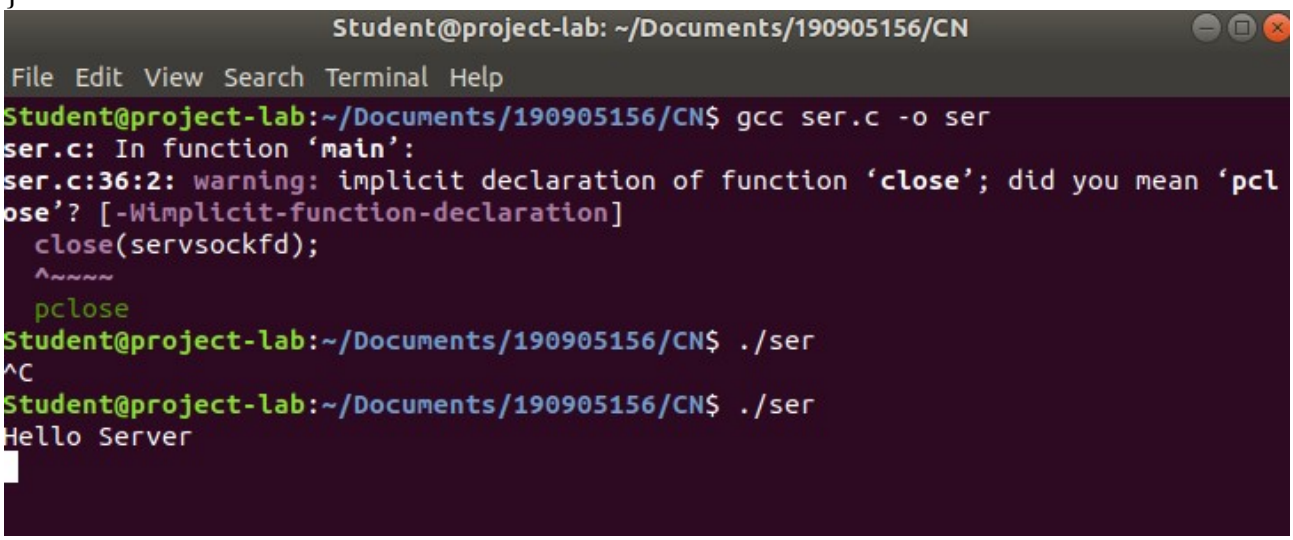
Q1) Implement a UDP server-client socket which allows communication beteween the 2 processes.

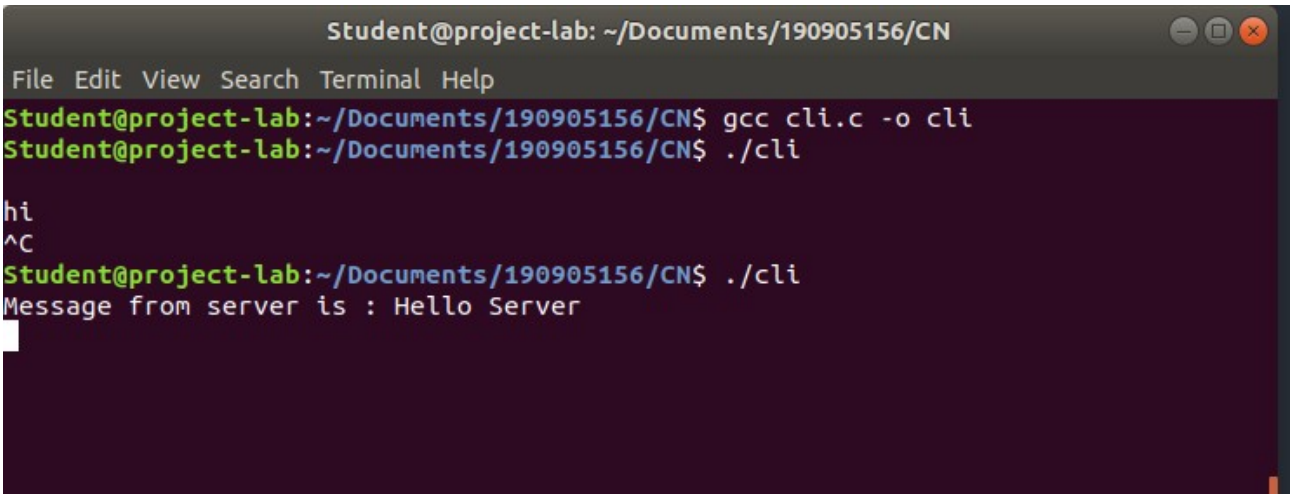
client.c

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define PORT 5000
#define MAXLINE 1000


int main()
{
        char buffer[100];
        int sockfd, n,len;
        struct sockaddr_in servaddr, cliaddr;

        char *message = "Hello Server";

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)&servaddr, sizeof(servaddr));
        len=sizeof(cliaddr);

        n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len );
   buffer[n]='\0';
        printf("Message from server is : %s \n",buffer);
   getchar();

        close(sockfd);
}
```

server.c

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

int main()
{
        char buffer[100];
        int servsockfd, len,n;
        struct sockaddr_in servaddr, cliaddr;
        bzero(&servaddr, sizeof(servaddr));


        servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;


        bind(servsockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));


        len = sizeof(cliaddr);
        n = recvfrom(servsockfd, buffer, sizeof(buffer),0, (struct sockaddr*)&cliaddr,&len);
        buffer[n] = '\0';
        puts(buffer);


   sendto(servsockfd, buffer, n, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
   getchar();

        close(servsockfd);
}
```





Q2) Implement a TCP server-client socket which allows communication beteween the 2 processes.

Client.c

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void clifunc(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;
		}
	}
}
```

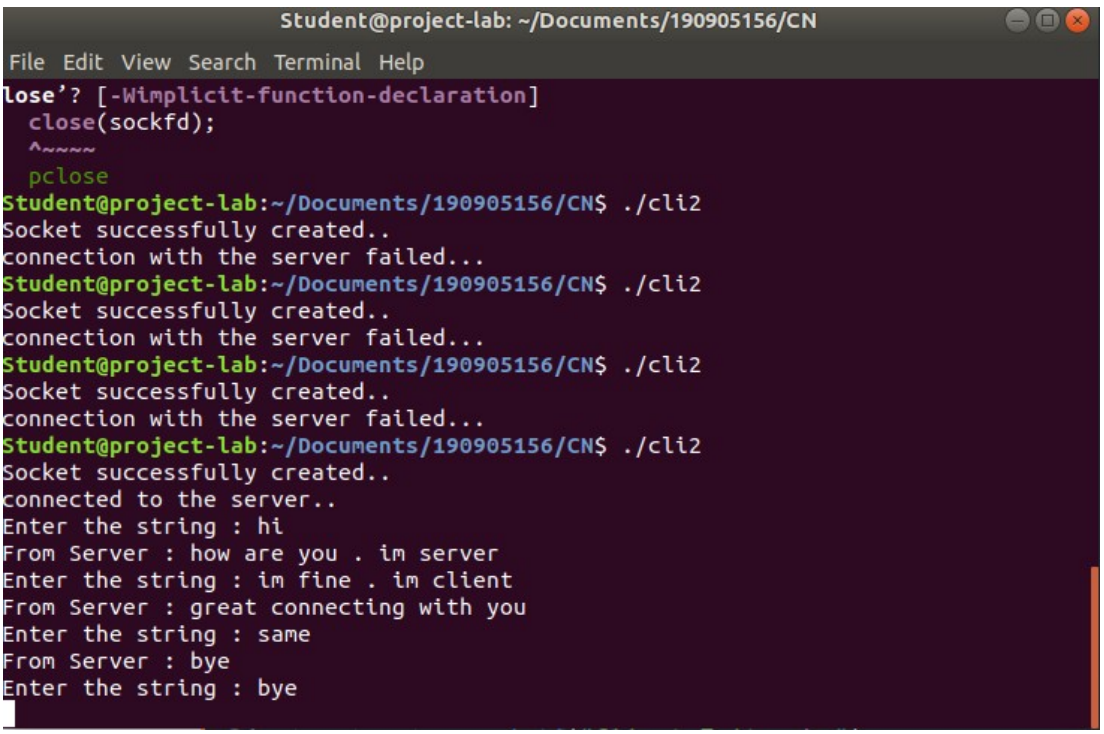
```
int main()
{
	int sockfd, connfd;
	struct sockaddr_in servaddr, cli;

	// 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 client
	clifunc(sockfd);

	// close the socket
	close(sockfd);
}
```

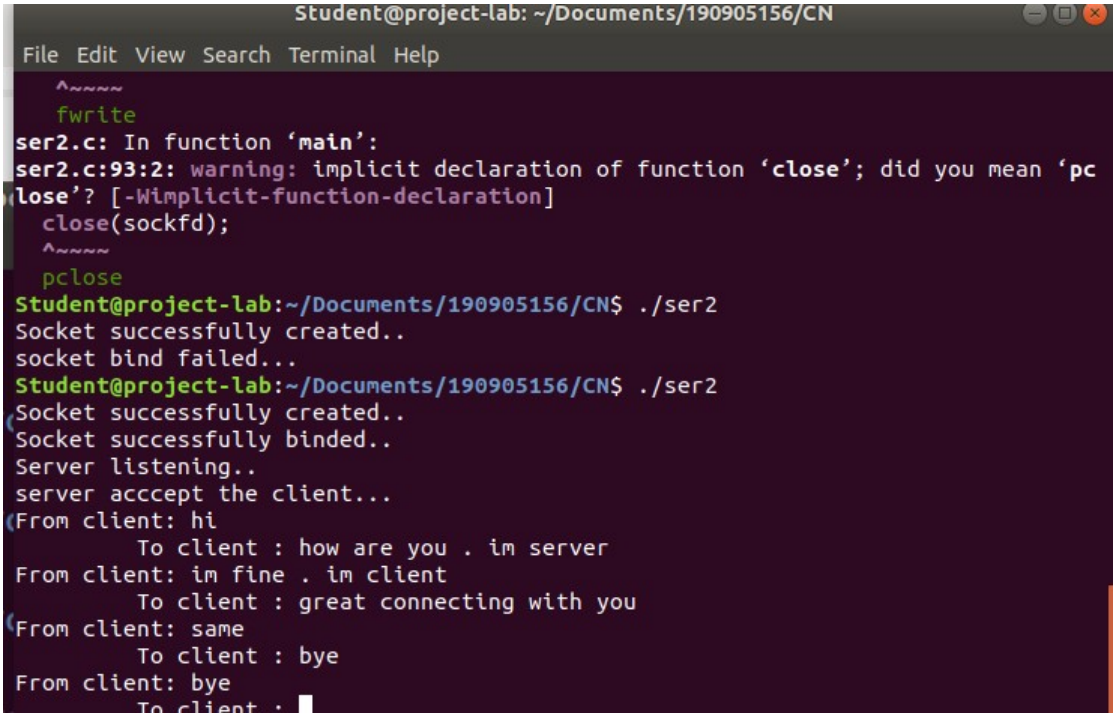
server.c

```
#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 servfunc(int sockfd)
{
	char buff[MAX];
	int n;
	// infinite loop for chat
	for (;;) {
		bzero(buff, MAX);

		// read the message from client and copy it in buffer
		read(sockfd, buff, sizeof(buff));
		// print buffer which contains the client contents
		printf("From client: %s\t To client : ", buff);

		bzero(buff, sizeof(buff));
// Read server message from keyboard in the buffer
		n=0;
		while ((buff[n++] = getchar()) != '\n')
			;
// and send that buffer to client
		write(sockfd, buff, sizeof(buff));

		// if msg contains "Exit" then server exit and session 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 acccept failed...\n");
        exit(0);
    }
    else
        printf("server acccept the client...\n");

    // Function for chatting between client and server
    servfunc(connfd);

    // After sending exit message close the socket
    close(sockfd);
}
```




Student@project-lab: ~/Documents/190905156/CN
File Edit View Search Terminal Help
^~~~~
fwrite
ser2.c: In function ‘main’:
ser2.c:93:2: warning: implicit declaration of function ‘close’; did you mean ‘pc
lose’? [-Wimplicit-function-declaration]
close(sockfd);
^~~~~
pclose
Student@project-lab:~/Documents/190905156/CN$ ./ser2
Socket successfully created..
socket bind failed...
Student@project-lab:~/Documents/190905156/CN$ ./ser2
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: hi
To client : how are you . im server
From client: im fine . im client
To client : great connecting with you
From client: same
To client : bye
From client: bye