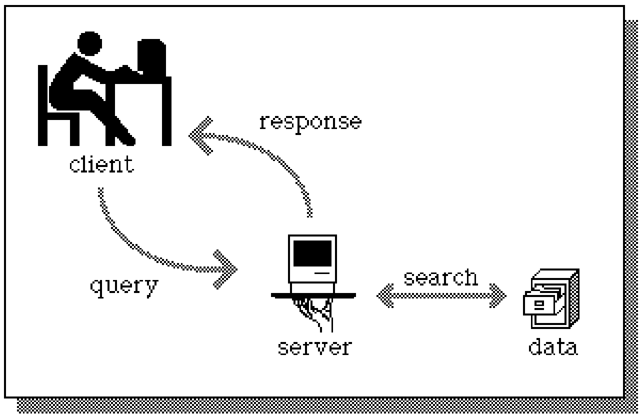
TRIBHUVAN UNIVERSITY

INSTITUE OF ENGINEERING

PULCHOWK CAMPUS

Pulchowk, Lalitpur



A Report on C Programming

"Client - Server Application in C"

<u>SUBMITTED BY:</u>

Ashok Basnet (066/BCT/505)
June 14, 2010

<u>SUBMITTED TO:</u>

Department of Electronics
&
Computer Engineering

Acknowledgement

Although, this project was developed individually, it would not have been possible without the help of my fellow friends and teachers. I gratefully acknowledge the kind help, support and guidance provided by Mr. Anish B. Amatya in the project and his valuable suggestions for the project. I would like to thank Department of Electronics and Computer for including the mini project concept in the Bachelor of Engineering I/I.

This project would not have got the success without the co-operation of the teachers and friends and I would like to thank all of them for their help and suggestions for the improvement. I would like to thank Mr. Pradip Maharjan for his valuable suggestion in use of socket in the program.

Abstract

Networking has become an integral part of our everyday life and we are in some way connected to others. It has revolutionized the way we are living. Many times we need our application to write some data into a file, stored on a remote machine connected through a network, display some message in desktop of remote machine or simply exchange messages across the machines connected to the network. This project is intended to be used as the server side system which can manage the student information and host the server to accept the connections from the clients across network. When the application is run at client mode, it can access the information of server by providing the required roll number of the student and chat with the server.

Table of Contents

Objectives

The objective behind developing the client server application in C is:-

1. To have the knowledge of C and implement it in the development of customized software.
2. To know the basic file operation and its handling in C.
3. To be familiar with the client server architecture.
4. To know how the network communication occurs whether be in the local area network to the World Wide Web (www).
5. To get concept of sockets and use it to communicate between computers within network.
6. To make an application which can get client request and reply back with the required results.
7. To manage the information of students and make them available to the other people in network if required.

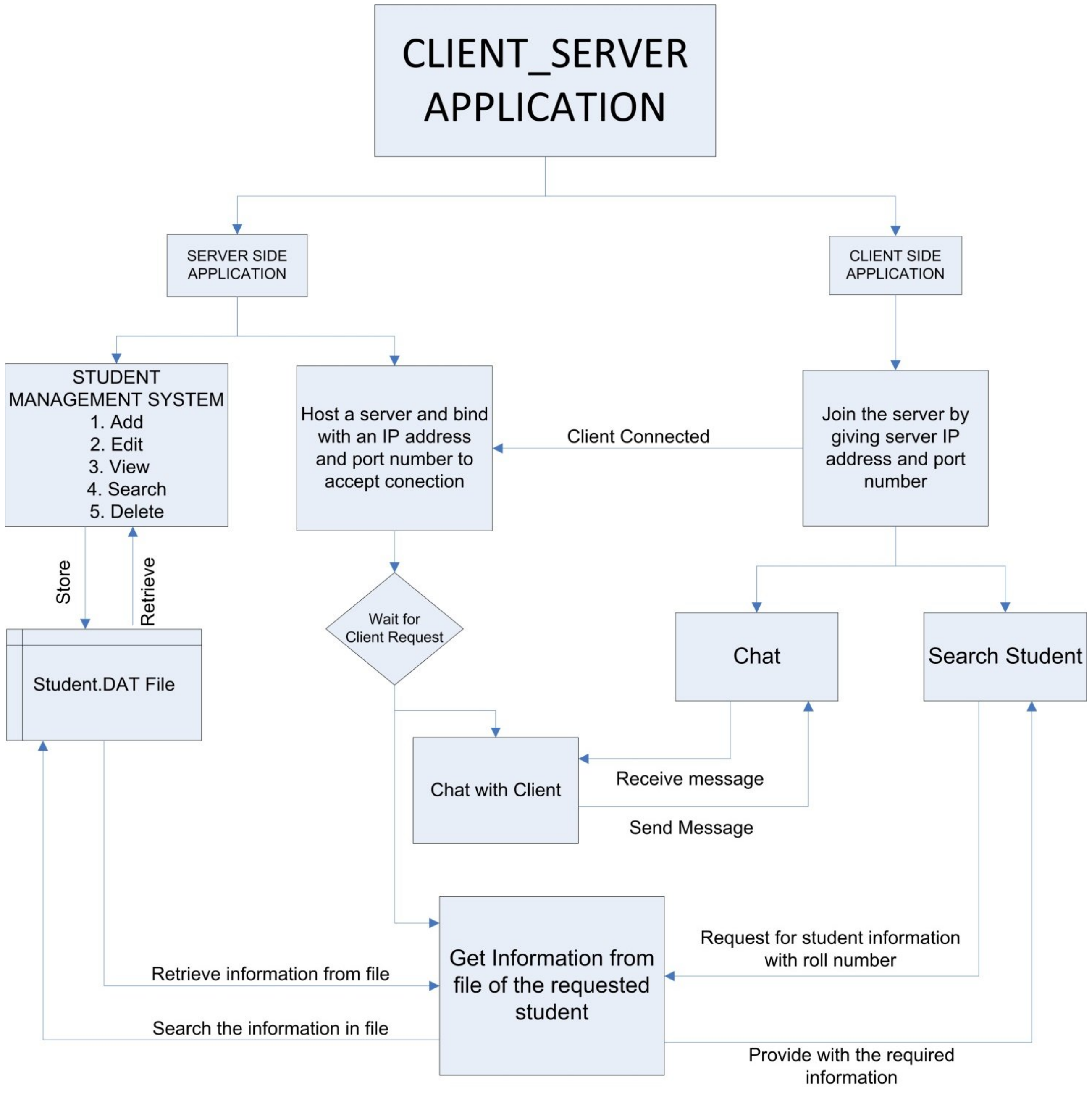
Introduction

The project is based on the client server architecture and its communication protocols. The project basically is divided into two sections:- Server and Client. The server has the full control of the information. The server can manage the information of the students and host the server to listen to clients so that client can get required information.
The features of the project include client which can chat with server for any reason or can search the information of a particular student. The chat part basically deals with message sending and receiving between server and client. When a client requests for the chat, server starts the session of chat and both can send and receive the message one at a time. When the chat session needs to be closed any user can type “end” and send it over network.
The server can store the information of the students and do all sorts of modification, search, delete them. The information is stored in the file at server. When client requests for it, the server searches the required from the file and replies back to the client.

This project is an implementation of how the information is exchanged in any network using sockets. Windows sockets 2 (winsock) is used to create application capable of transmitting data across network independent of the network protocol being used. Winsock follows the Windows Open System Architecture (WOSA) model; it defines a standard service provider interface (SPI) between the application programming interface (API), with its exported functions and the protocol stacks. It uses the sockets paradigm that was first popularized by Berkeley Software Distribution (BSD) UNIX. It was later adapted for Windows in Windows Sockets 1.1, with which Windows Sockets 2 applications are backward compatible. Winsock programming previously centered around TCP/IP. Some programming practices that worked with TCP/IP do not work with every protocol. As a result, the Windows Sockets 2 API adds functions where necessary to handle several protocols.

Program Details

The program is divided into two sections:- Server and Client. The application can standalone act as either Server or Client. The flow of program can be better explained with the diagram below:-


CLIENT_SERVER APPLICATION
SERVER SIDE APPLICATION
CLIENT SIDE APPLICATION
STUDENT MANAGEMENT SYSTEM
1. Add
2. Edit
3. View
4. Search
5. Delete
Host a server and bind with an IP address and port number to accept conection
Client Connected
Join the server by giving server IP address and port number
Store
Retrieve
Student.DAT File
Wait for Client Request
Chat
Search Student
Chat with Client
Receive message
Send Message
Get Information from file of the requested student
Request for student information with roll number
Retrieve information from file
Search the information in file
Provide with the required information

Theory

C Programming Basics

Structure

C is provided with a constructed data type known as structures, a mechanism for packing data of different types. A structure is a convenient tool for handling a group of logically related data items. For example, it can be used to represent a set of attributes, such as student _ name, roll_ number and marks. The concept of a structure is analogous to that of a 'record' in many other languages.
Structures help to organize complex data in a more meaningful way. It is a powerful concept that we may often need to use in our program design. This chapter is devoted to the study of structures and their applications in program development.
In general, the syntax for structure definition is:

```
              struct struct_name
           {
                 data _ type mem1;
               data _ type mem2;
             ……………………….
             ……………………….
            data  _type memn;
        };
```

The structure definition starts with keyword struct followed by an identifier or tag name. The tag name is structure name and can e used for instantiating structure variable. In above definition, struct _name is referred to as structure name or structure tag name; and mem1, mem2, memn are known as structure members or elements or fields. These members are enclosed within braces and terminated by semicolon.

After the structure has been specified, the structure variable can be declared as standard data type:

```
 struct struct_name var1, var2,………,varn;
```

The structure definition serves as a template for user defined data type. It does not reserve memory unless a variable of structure data type is declared.

Accessing Member of structures

In C programming language, the members of a structure are processed individually as separate entities. We make use of period or dot "." Operator to access the individual members of structure.The syntax for accessing member of a structure variable is follows:

```
          struct _variable. member
```

Where struct _ variable refers to the name of a structure variable, and member refers to the name of member within the structure. The dot (.) is an operator that separates the variable name from the member name. We noticed that the dit operator must have precedence among all operators and has left to right associatively. Consider the following statement:

```
    struct  employee e1;
```

Now, each member of the structure variable e1 can be accessed using the dot operator as follows:

e1.emp_id
The employee’s employee IDnumer is accessed;
e1.name
The employee’s name is accessed. e1.salaryThe employee’s salary is accessed.

File Handling

The console function like printf() and scanf() have been used for input/output .This scheme is adequate if the volume of data involved in not so large or it is not necessary to store the information for further use. However, many applications may require a large amount of data to be read, processed, and also saved for later use. Such information is stored on the auxiliary memory device in the form of data file.And a file is a collection of bytes that is given a name. In most computer systems, files are used as a unit of storage primarily on floppy-disk or fixed-disk data storage system (or they can be CDs or other storage device). Thus data files allow us to store information permanently, and to access and alter that information whenever necessary.

The file handling function available in standard library in order to implement I/O midel is classified as follows:

a) File access
b) Operation input/output
c) Formatted input/output
d) Character input/output
e) Direct input/output
f) File positioning
g) Error handling

The file access included the function like fopen() to open a file, fclose() to close a file , fflush () to flush out the buffer associated with a file, and freopen() to change the file associated with a stream. Also setvbuf() and setbuf() functions are use to allow the users explicitly control the file buffering strategy.

The operation on file includes like remove() to remove a file, remname() to rename a file ,tempfile() to create a temporary binary file and tmpnam() to generate a unique filename.

Formatted input/output group includes the n functions fscanf(), scanf() and sscanf() to read formatted data. Similarly fprintf() ,printf(), sprint(), vfprintf(), vprintf() and vsprintf() to write formatted data.

The character input/output group includes the functions fgetc() ,getc() and getchar() to read a character from an input stream and functions ungetc() to push back a character to an input stream. The functions fgets() and gets() are to read strings and the output functions fputc(),putc(), putchar(), fputs() and puts() are also included in this group.

The direct input/output group includes functions fread() to read and fwrite() to write a certain number of data items specified size.

File positioning group includes functions fread() to read and fwrite() to write a certain number if data items specified size.

File positioning group includes functions to set the file positon to some specified value to allow access to a specific portion of the seek(),interrogate the current file position ftell(),and reset the file position to the beginning of the file rewind().

Error handling group include functions to test whether EOF returned by a function indicates an end-of-file or an error (feof and ferror), clear end-of-file and indicators clearer, and map the error number errno to an error message perror.

Different Modes for opening a file:-

S.N	Mode	Meaning
1	"r"	Open a text file for reading
2	"w"	Create a text file for writing
3	"a"	Append to a text file
4	"rb"	Open a binary file for reading
5	"wb"	Open a binary file for writing
6	"ab"	Append to binary file
7	"r+"	Open a text file for read/write
8	"w+"	Create a text file for read/write
9	"a+"	Append or creat a text file for read/write
10	"r+b"	Open a binary file for read/write
11	"w+b"	Create a binary file for read/write
12	"a+b"	Append a binary file for read/write

Computer Networting

A computer network is group of computers that are connected together to share resources, such as hardware, data, and/or Software. It is a data communication system that interconnects computer Systems at different sites. Networking is a collection of individual networks, connected by Intermediate Working devices that function as a single large network. Industries, colleges & many business organizations use internetworking. For many purpose such as fast communications internally, sharing a device for many computers and for many other purposes also.

Traditional telecommunication links transmit data as a series of bytes, characters, or bits alone. Unlike this, in a computer network data is transmitted in the form of packets. Once the data is formatted into a packet, the network can transmit longer messages more efficiently and reliably. A packet consists of three elements- header, payload and trailor. As the names suggest, header and trailor are used to mark the beginning and end of packet, whereas the payload the actual data that is to be transmitted.

Protocols

A protocol is a set of rules that governs the communications between computers on a network. These rules include guidelines that regulate the following characteristics of a network: access method, allowed topologies, types of cabling, and speed of data transfer.

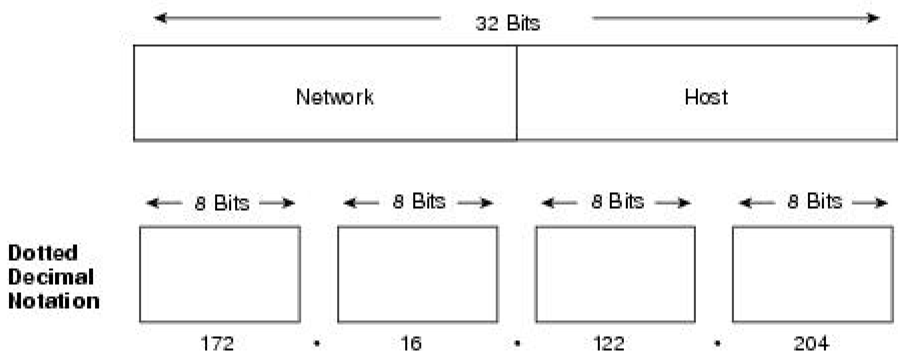
Network communication is very complex. It involves taking decision about how to create packets, how to detect and correct errors in transmission of data, which path to use for sending packets from one machine to another, how to support multiple OS, how to deal heterogeneous networking cabling etc. To make this complexity more manageable, the different aspects of network communication is divided into layers. Each layer can use different protocols. Common layers are:-

a. Application Layer- HTTP,SMTP,POP3,FTP
b. Transport Layer- TCP,UDP
c. Network Layer – IP,ICMP
d. Physical Layer – Ethernet

IP Addressing

The IP addressing scheme is integral to the process of routing IP data gram through an internet work. Each IP address has specific components and follows a basic format. These can be sub divided and used to create addresses for sub networks. Each host on a TCP/IP network is assigned a unique 32-bit logical address that is divided into two main parts: the network number and the host number. The network number identifies a network and must be assigned by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet

The 32-bit IP address is grouped eight bits at a time, separated by dots, and represented in decimal format (known as *dotted decimal notation*). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet is 0, and the maximum value for an octet is 255.



Port Numbers

A port number is a way to identify a specific process to which an Internet or other network message is to be forwarded when it arrives at a server. For the Transmission Control Protocol and the User Datagram Protocol, a port number is a 16-bit integer that is put in the header appended to a message unit.
In computer networking, a port number is part of the addressing information used to identify the senders and receivers of messages. Port numbers are most commonly used with TCP/IP connections. Home network routers and computer software work with ports and sometimes allow you to configure port number settings. These port numbers allow different applications on the same computer to share network resources simultaneously. In both TCP and UDP, port numbers start at 0 and go up to 65535. Numbers in the lower ranges are dedicated to common Internet protocols:-

HTTP – 80	SMTP – 25	POP3 – 110
FTP – 20,21	Time – 37	Telnet – 23
Whois – 43 etc.		

Byte Ordering

Different types of machines use different byte orders. Byte ordering or Endianess is the attribute of a system which indicates whether integers are stored / represented left to right or right to left.

Example : int x = 0xAABBCCDD
This 4 byte long integer can be represented in the same 2 orderings:
Big Endian:

```
Byte Value: [0xAA]  [0xBB] [0xCC] [0xDD]
Memory:     [  0   ] [  1   ] [  2  ] [  3  ]
```

Little Endian:

```
Byte Value: [0xDD]  [0xCC] [0xBB] [0xAA]
Memory:     [  0   ] [  1   ] [  2  ] [  3  ]
```

All Network data is sent in Big Endian format. In the networking world we call this representation as Network Byte Order and native representation on the host as Host Byte Order. We convert all data into Network Byte Order before transmission.

Some utility functions:

- *Byte Ordering:*
  Host Byte Order to Network Byte Order:
  htons() , htonl()
  Network Byte Order to Host Byte Order:
  ntohs() , ntohl()
- *IP Address format*:
  Ascii dotted to Binary: inet_aton()
  Binary to Ascii dotted: inet_ntoa()

The Client – Server model

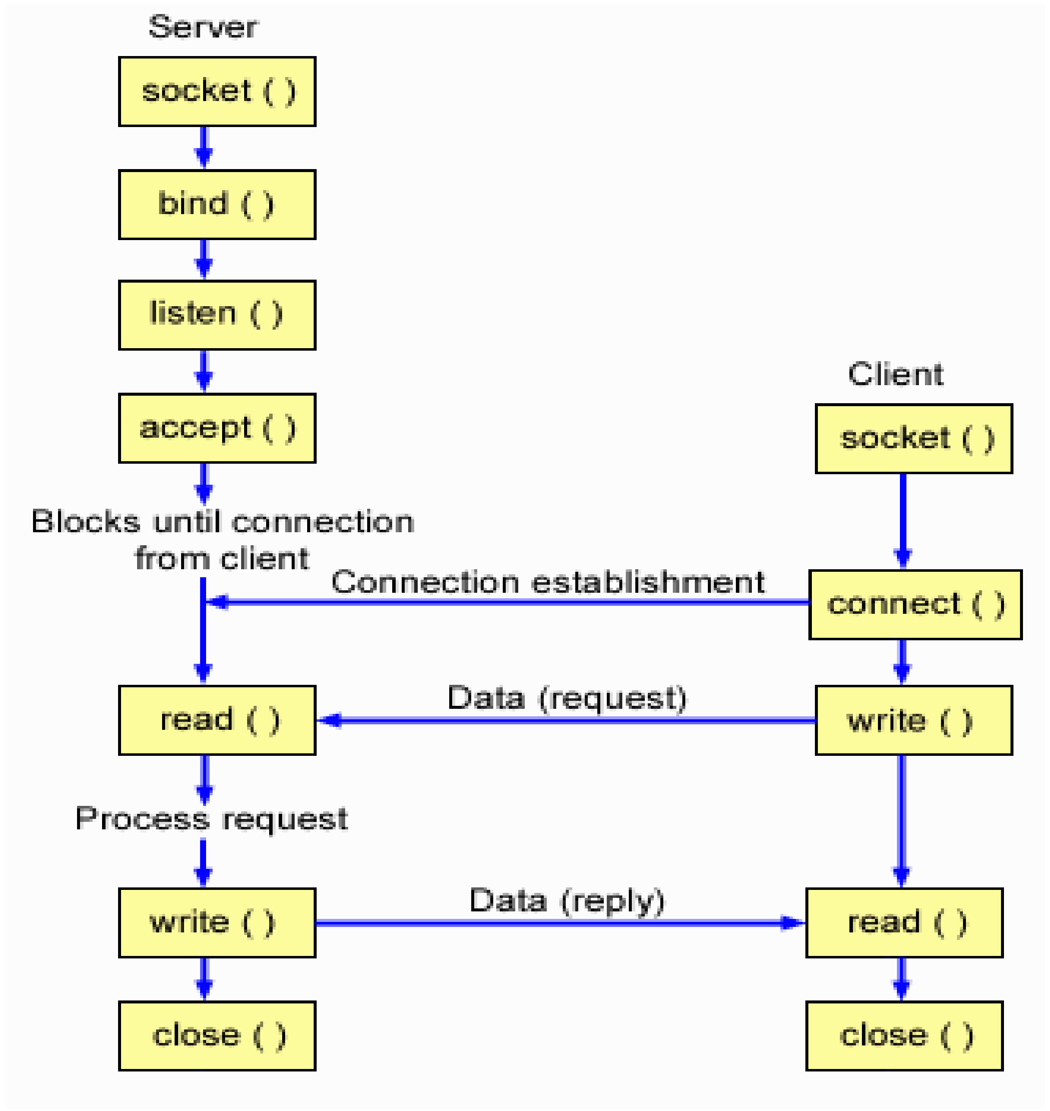
- Server – An entity which is a provider of information.
- Client – An entity which is a seeker of information.
- Example – Apache is a web server providing web pages (information) and Internet Explorer is a web client which requests those pages from the server.
- In the socket programming world almost all communication is based on the Client-Server model.
- The Server starts up first and waits for a client to connect to it. After a client successfully

connects, it requests some information. The Server serves this information to the client. The client then disconnects and the Server waits for more clients.

Sockets

Sockets are network communication channels. Socket is a protocol independent method of creating a connection between processes. They are the end points for the communication channel. A socket is used by applications as an interface to the underlying network and protocols. Applications that communicate with one another in a network carry out the communication using sockets. Windows provides a set of API functions for creating sockets and sending/receiving packets through them. This set of functions is commonly known as Winsock API.

A TCP Server – Client Interaction



The first calls the socket which creates an endpoint for the communications. The client also does the same by calling socket. The server then calls bind() to attach an unique IP address and a port number to which it is going to be listening for various clients. Then the server goes to accept stage where it waits for a client to connect to it. At that time the client will use the connect () call to establish a connection to the server. Once the client calls connect() and the connection has been established with the server both of them can call read() or recv() and write or send() to transfer data among each other. Once the data has been transferred they call the close function to close the connection.

Functions of Socket Programming or syscalls()

These are the standard functions for the socket programming :

a) socket()
b) bind()
c) listen()
d) accept()
e) connect()
f) send()
g) recv()
h) closesocket()

1. socket() – A Connection Endpoint

- This creates an endpoint for a network connection.
  Syntax:
  Int Socket(int doman, int type, int protocol)

  domain = AF_INET (IPv4 communication)
  type = SOCK_STREAM (TCP) , SOCK_DGRAM (UDP)
  protocol = TCP/IP or UDP
- Example : socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  This will create a TCP socket.
- The call returns a socket descriptor on success and -1 on an error.

2. bind() – Attaching to an IP and Port

   A server process calls bind to attach itself to a specific port and IP address.
   Syntax:
   Int Bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen)

   sockfd = socket descriptor returned by socket()
   my_addr = pointer to a valid sockaddr_in structure cast as a sockaddr * pointer
   addrlen = length of the sockaddr_in structure

4. listen() – Wait for a connection
   The server process calls listen to tell the kernel to initialize a wait queue of connections for this socket.
   Syntax:
   Int Listen(int sock, int backlog)

   sock = socket returned by socket()

backlog = Maximum length of the pending connections queue

5. accept() – A new connection
   Accept is called by a Server process to accept new connections from new clients trying to connect to the server.
   Syntax:
   Int Accept(int socket, (struct sockaddr *)&client, socklen_t *client_len)

   socket = the socket in listen state
   client = will hold the new client's information when accept returns
   client_len = pointer to size of the client structure

6. connect() – connect to a service
   Connect is called by a client to connect to a server port.
   Syntax:
   Int Connect(int sock, (struct sockaddr *)&server_addr, socklen_t len)

   sock: a socket returned by socket()
   server_addr: a sockaddr_in struct pointer filled with all the remote server details and cast as a sockaddr struct pointer
   len: size of the server_addr struct

7. send / recv – Send and receive data across network
   Send(), Recv calls are used to send and receive data .
   Syntax:
   Int send(int sock, void *mesg, size_t len, int flags)

   Int recv(int sock, void *mesg, size_t len, int flags)

   sock = A connected socket
   mesg = Pointer to a buffer to send/receive data from/in .
   len = Size of the message buffer
   flags = 0

   The return value is the number of bytes actually sent/received.

8. closesocket() – Close the socket
   Close signals the end of communication between a server-client pair. This effectively closes the socket.
   Syntax:
   Int close(int sock)
   sock = the socket to close

Algorithms

Main Program Functioning

1. Start
2. Create necessary variables required for the project
3. Display welcome screen.
4. Display the main menu and take input from user as the choice
5. Switch case the options provided.
6. Display the close screen
7. End

Communication Steps for Client and Server

A network application works as follows:-

1. An endpoint for communication is created on both ends.
2. An address is assigned to both ends to distinguish them from the rest of the network.
3. One of the endpoints initiates a connection to the other.
4. The other end point waits for the communication to start.
5. Once a connection has been made, data is exchanged.
6. Once data has been exchanged the endpoints are closed.

Server

1. Initialize Winsock.
2. Create a socket.
3. Bind the socket.
4. Listen on the socket for a client.
5. Accept a connection from a client.
6. Receive and send data.
7. Disconnect.

Client

1. Initialize Winsock.
2. Create a socket.
3. Connect to the server.
4. Receive and send data.
5. Disconnect.

Functions used in the program

a) open_file()

The function is used to open a binary file student.DAT to either 'rb+' mode or 'wb+' mode according if it is already created or is being created for first time.

Syntax:

```
void open_file();
```

```
Algorithm:-
1. Open a file student.DAT in ‘rb+’ mode with file pointer fp.
2. Is fp == NULL
        Yes:  Open the file student.DAT in ‘wb+’ mode
              call the function open_file
        Is fp == NULL
              Yes: Display “Cannot open file”:
                   Exit Program
3. Exit
```

b) count_records()

This function is used to count the number of records in a file.

Syntax:

int count_records();

```
Algorithm:-
1. Open a file student.DAT in ‘rb+’ mode with file pointer fc and set counter variable to 0.
2. set the file position cursor to the beginning of file.
3. Loop until ‘End of File’ is not encountered
       Increase the counter by 1

4. Close file
5. Return counter
```

c) checkEmpty()

The function checks the input from user and return the 0 or 1 according if matched with the NOTMODIFY variable ‘#’ in this case. It is used in the edit function where if the user don’t want to modify or change the current value of the information, he/she can type simply ‘#’ so that it won’t be modified.

Syntax:

int checkEmpty(char *chkStr);

returns 0 if the input from the user is equal to the NOTMODIFY value

```
Algorithm:-
1. Check the input string from the function chkStr with the NOTMODIFY constant.
2. Is strcmp(chkStr,NOTMODIFY) equals 0
        Yes: return 0
        No : return 1
3. Exit
```

d) checkRoll ()

The function checks if a record with the given roll number is already present in the file so that the replication of the record is probihited.

Syntax:

int checkRoll (char *str);

returns 1 if the requird record is already present in the file.

Algorithm:-

```
1. Open a file student.DAT in 'rb+' mode with file pointer fc and set counter variable
to 0.
2. Set the file position cursor to the beginning of file.
3. Loop until 'End of File' is not encountered
        a. Compare strings str with the roll number of structure variable
           that is extracted from file.
        b. Is result == 0
              Yes: return 1
6. Close file
7. Return 0
```

e) gotoxy ()

The function puts cursor to specific postion in the console(x and y). Actually this function was by default the Turbo C++ function, but for Code::Blocks it has to be userdefined.

Syntax:

void gotoxy (int x,int y);

Algorithm:-

```
1.  Define a global variable  COORD coord = {0, 0};
2. coord.X = x and coord.Y = y
3. Call the function
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),      coord);
1.      Exit
```

f) intToString ()

The function converts the integer to string. This was needed for the send function of the socket programming where the marks of students needs to be send across the network in string format.

Syntax:

char* intToString(int n, char str[]);

Algorithm:-

```
1. Set i, negative, temp to 0
2. Is n negative
      Yes: Make it positive
           negative = n<0
3. Generate digit characters in reverse order
```

```
a. Create a rightmost digit
      str[i++] = '0'+n%10;
b.  Remove the digit
      n /= 10;  c.
c.   Is n>0
      Yes: goto step 3
   No: goto step 5

5. Create a rightmost digit
6. If it was negative
      Append minus
8.   Append terminator i.e. null character for the end of string
9.   Now reverse the string using strrev function
10. Return str
```

6. socket_start()

   This function initializes windows socket and creates a socket for a connection.

   Syntax:

   void socket_start()

   Algorithm:-

```
1. Create a WSADATA called wsadata to initialize the use of  ws2_32.dll.
2. Call WSAStartup and return its value as an integer and check for errors.
   Is the returned value is not zero
         Yes: a) Display “Socket Initialization: Error with WSAStartup,
                  required version snot available ".
              b) Release the resources and exit from program.
3. Create a Socket with IPv4 address family and  TCP/IP protocol
4.Is Socket a INVALID_SOCKET
         Yes: a) Display “Socket Initialization: Error Creating socket\\
              b) Release the resources  and exit from program.
         No: Display “Socket Initialised”
5.Exit
```

7. start_server()

   This function starts server, bind port, listen to incoming connections and accept connection with the client so that the exchange of the information between client and server.

   Syntax:

   void start_server()

   Algorithm:-

```
1. Create a server_address, a structure variable of sockaddr_in to store the   server
configuration.
2. Bind the socket created to attach an IP address and port number to server.
   Is the returned value SOCKET_ERROR.
         Yes: a) Display “Failed to bind the socket”
```

```
                    b) Release the resources and exit from program.
                   No: Display ”Socket binded”
3. Listen or wait for connection
   Is the returned value SOCKET_ERROR.
           Yes: a) Display “Error listening on socket”
                    b) Release the resources and exit from program.
                   No: Display ”Accepting Connections”
4. Retrieve the client information and store it in client structure .
5. Accept connection from the client
     Is SOCKET_ERROR
           Yes: Display “Couldn’t accept connections” and exit
                   No: Display ”Accepted a Connection”

6. Start the socket for the client for connection
7. grab ip address of client and stores that in form of string
8. display the ip address of the client that is being connected
9. Exit.
```

8. send_data ()

   This function is used to send the data across in the fom of string.

   Syntax:

   bool send_data(char *buffer);

   Algorithm:-

```
1. Take input as a string that needs to be send across the network to client.
2. Send the data
       Is the error SOCKET_ERROR encountered
               Yes: set the variable done  =  true
3. return true
```

9. recv_data ()

   This function is used to receive the data from the network and store it in the required string.

   Syntax:

   bool recv_data(char *buffer);

   Algorithm:-

```
1. Receive the data from the network.
2. Attach a null value after received the message.
3. Return true
```

10. GetAndSendMessage ()

   This function is used to get and send the message across the network

   Syntax:

   void GetAndSendMessage();

   Algorithm:-

```
1. Get the required string that has to be send across the network.
2.Is the sring “end”
```

```
            Yes: Set done = true
    3. Send the message by calling the function send_data.
    4. Exit
```

11. server_char ()

This function is an interface for the chatting with client. It sends and receives the chat message across the network. The function runs until any one side sends a message “end”

Syntax:

void server_chat ();

Algorithm:-

```
    1. Set done = false.
    2. Display “Client is typing”
    3. Receive the message from network
    4. call the function time(&t) for the current system time
    5. Display the received message and time in which the message is receieved.
    6. Call the function GetAndSendMessage to send the message from server
    7. Is the received string “end”
            Yes: done = true
    8. Is done not equal to true
            Yes: goto step 2
    9. Call the function host_server after chat session is over
    10. Exit
```

12. client_chat ()

This function is an interface for the chatting with server. It sends and receives the chat message across the network. The function runs until any one side sends a message “end”

Syntax:

void client _chat ();

Algorithm:-

```
    1. Set done = false.
    2. .Call the function GetAndSendMessage to send the message from server
    3. Display “Server is typing”
    3. Receive the message from network
    4. Call the function time(&t) for the current system time
    5. Display the received message and time in which the message is receieved.
    7. Is the received string “end”
            Yes: done = true
    8. Is done not equal to true
            Yes: goto step 2
    9. Call the function join_server after chat session is over to re-request the service if
    any.
    10. Exit
```

13. request_info ()

This function is used by client to request for the student information on server.

Syntax:

```
void request_info ();
```

Algorithm:-

```
1. Set done = false.
2. Get the roll number of the student from user
3. Send the roll number across the network
4. Receive the message from server as response to record
      Is received message NOTFOUND
            Yes: a) done = TRUE
                 b)Display "Record Not Found"
            No: Display "Record Found"
5. Loop while done is not equal true
            a) Receive message from server
            b) Is received message equals "end"
                  Yes: done = true
                  No: Display required information of student.

6. Does the user wants to search another record again
            Yes: go to step 1
7. Call the function join_server to again join server for information
8. Exit
```

14. give_info ()

This function is used by server to give the information required by the client.

Syntax:

```
void  give_info ();
```

Algorithm:-

```
1. Set done = false.
2. Open the file using open_file function in rb+ mode.
3. Receive the message from network sent by client.
4 Assign the pointer to the beginning of the file to be read
5. Loop until 'End of file' is not encountered read data from file.
6. Is received roll number equals roll number on a file
            Yes: send data "FOUND" and along with it all the associated
      information related to it.
7. Does user sends no for next record
            Yes: go to step 8
            No: go to step  1
8. Call the function host server
10. Exit
```

15. join_server ()

This function is called by the client in order to join the server and request the required information.

Syntax:

```
void  join_server ();
Algorithm:-
1. Is SOCKET_START is false
        Yes: a) SOCKET_START = true
             b) Call the function start_socket to start the socket for client
             c) Call the function connect_server to connect with the server
2. Get the service that client wants from sever as options
        a) Chat
        b) Search Student Info
        c) Go to main
3. Is
    Option equals 1, call function client_chat to start chat
    Option equals 2, call the function request_info to search the student.
    Option equals 3, call the function main_menu to return back to main.
    Else request for the option again.
4. Exit
```

16. host_server ()

This function is called by the server in order start the server and takes request from the user.

Syntax:

```
void host_server ();
Algorithm:-
1. Is SOCKET_START is false
        Yes: a) SOCKET_START = true
             b) Call the function start_socket to start the socket for server
             c) Call the function start_server to start the server
2. Display "-- --Waiting for Client Request ----".
3. Receive the service that client want from sever as options
        a) Chat
        b) Search Student Info
3. Is
    Option equals 1, call function server_chat to start chat
    Option equals 2, call the function give_info to give information.
    Else call the function host_server
4. Exit
```

17. add_student ()

This function is used to add the student information.

Syntax:

```
void add_student ();
Algorithm:-
1.  . Count the number of records in file using count_records function and add
to get the current student Id
```

```
1. Open a file using open_file function.
2. Assign the pointer to the end of the file to write
3. Get current date and time for regDate.
4. Get data from user about student and store it in student structure
5. Close file
6. Does user wants to add more books
        Yes: Go to step 1
        No: Call the function main_menu
7. Exit
```

18. view_student ()

This function is used to view all the students registered.

Syntax:

void view_student ();

Algorithm:-

```
1. Count the number of records in file using count_records function to get total number of students
2. Open a file using open_file function.
3. Assign the pointer to the beginning of the file to read from beginning
4. Loop until ‘End of file’ is not encountered read data from file.
        a) Display the student information.
5. Show all the list of students.
6. Exit
```

19. search_student()

This function is used to search student with roll number as primary Id.

Syntax:

void search_student ();

Algorithm:-

```
1. Get roll number from user to search.
2. Open a file using open_file function.
3. Assign the pointer to the beginning of the file to read from beginning
4. Loop until ‘End of file’ is not encountered read data from file.
        a) Is roll number entered from user equals to that in file.
                Yes: Display all the related information.
5. Exit
```

20. edit_student ()

This function is used to search student with roll number as primary Id.

Syntax:

void edit_student();

Algorithm:-

```
1. Get roll number from user to search.
2. Open a file using open_file function.
3. Assign the pointer to the beginning of the file to read from beginning
```

```
4. Loop until 'End of file' is not encountered read data from file.
      a) Is roll number entered from user equals to that in file.
              Yes: a) Display all the related information.
                   b) Prompt user to add new record
                   c) Does the user types '#' for not changing the current info
                        Yes: Do not modify the current record information
                              No: Set the new value to the record info
                   d) Move file pointer to the previous record end by using fseek
                   e)  Write the modified record into file
                   f)  Close the file
5. Add another student
              Yes: go to step 1
              No: go to student menu
6. Exit
```

21. delete_student ()

This function is used to delete student record.

Syntax:

```
void  delete_student();
```

Algorithm:-

```
1. Get roll number from user to delete.
2. Open a file using open_file function.
3. Open a temporary file TEMP.DAT in write mode.
4. Assign the pointer to the beginning of the file to read from beginning
5. Loop until 'End of file' is not encountered read data from file.
      a) Is roll number entered from user equals to that in file.
              Yes:  Display all the related information
              No: Write records to temporary file
6. Prompt for sureness to delete
      Is he sure?
              Yes: a) Close the files student.DAT and TEMP.DAT
                   b) Remove student.DAT file
                   c) Rename TEMP.DAT to student.DAT.
7. Delete another student
              Yes: go to step 1
              No: go to student menu
8. Exit
```

Source Code

The code is divided into several header files for the ease of function access and easy to program and debug.

var.h

This file contains all the variable & constant declarations, functions declarations.
It includes all the necessary header files required by the program.

```
/*
        File:- var.h
        Desc:- contains all the variable & constant declarations, functions declarations
*/

#include <stdio.h>  //Provides the core input and output capabilities of the C language.
#include <conio.h>  //for doing console input output
#include <ctype.h> /*Contains functions used to classify characters by their types or to
convert between upper and lower case in a way that is independent of the used character
set*/
#include <stdlib.h> /*For performing a variety of operations, including conversion, pseudo-
random numbers, memory allocation, process control, environment, signalling, searching,
and sorting.*/
#include <windows.h> /*defines a very large number of Windows specific functions that
can be used in C.*/
#include <string.h> //For manipulating several kinds of strings.
#include <time.h> //For converting between various time and date formats.
#include <winsock2.h>   // contains functions for socket programmming
#define NOT_FOUND "NOT_FOUND"
#define NUM_SUBJECTS 6  //No of subjects to make marks
#define PASSWORD "ioe"  //Password for doing server related functions
#define NOTMODIFY "#"  //charcter to be entered by user  if he/she wants to keep current
value
FILE *fp;
int flag_menu;//checks which menu is active  like 0- main menu ,1-server, 2- client
COORD coord = {0, 0}; // sets coordinates to 0,0
bool SOCKET_START = false;  //TRUE if socket is already initialised
bool login_server = false;  // TRUE if server is alerady logged in

time_t t;  // structure for accessing system time

//struture to hold the information of the students
struct student{
        char regDate[40];
         int id;
        char fName[20];
        char lName[20];
        char roll[15];
        char address[40];
        char email[40];
        char faculty[20];
        char phone[20];
        int marks[NUM_SUBJECTS];
        int percentage;
}s;

//Global variables
char subjects[][30] = {"Maths-I","Physics","Applied Mechanics","Computer
Programing","Basic Electrical","Engineering Drawing-I"};
char request_infos[10][20] = {"Registerd Date","Id","Roll No","First Name","Last
Name","Faculty","Address","Phone","E-mail","Percentage"};
int i;
int recsize = sizeof(s);

//Function declarations
void main_menu();  //Main interface of the system
void help();        //Help about the program
void add_student(); //Add the students
void search_student();// search studnet information according to roll number
void open_file();  //open a file for read or write
void delete_student(); //Delete a student records
void welcome_screen();  //Display Welcome message
void close_screen(); //Close the application

//Utility functions
char* intToString(int n, char str[]);  //convert number to string
int checkEmpty(char *); //check the input from user and return the 0 or 1 according if
matched
void gotoxy(int,int); // put cursor to specific postion in the console(x and y)
int count_records();  //Count the number of records in a file
int checkRoll(char *);  //check if roll number is already saved


//Server client related functions
void server();  //Server related functions and its options
void client();   //Cilent realted functions and its options
void client_chat();  //Chat side for the client
void server_chat();  //Chat side for server
void host_server();   /*Host a server with a unique IP and port number, initialize the socket,
bind connection and listen to clients*/
void join_server();   // Client side function for connecting to server
void request_info(); //requst for the information to server
void give_info();  //on request from client, reply back
void stMenu();  //menu to manage the student infomation
```

chatApp.c

This is the main program interface which contains the main function that controls the execution of the rest of program and gives a flow to the program.

```
#include "function.h"

int main()
{
   welcome_screen();
   main_menu();
   while(1){
      switch(getch()){
         case '1':
            if(flag_menu == 0)
               server();
            else if(flag_menu == 1)
               host_server();
            else if(flag_menu == 2)
               join_server();
            else if(flag_menu == 15)
               add_student();
            break;
         case '2':
            if(flag_menu == 0)
               client();
            else if(flag_menu == 1)
               stMenu();
            else if(flag_menu == 15)
               view_student();
            break;
         case '3':
            if(flag_menu==0)
               help();
            if(flag_menu == 1 || flag_menu == 2 )
               main_menu();
            else if(flag_menu == 15)
               search_student();
            break;
         case '4':
            if(flag_menu == 0)
```

```
                close_screen();
            else if(flag_menu == 15)
                edit_student();
            break;
        case '5':
            if(flag_menu == 15)
                delete_student();
            break;
        case '6':
            if(flag_menu == 15)
                server();
            break;
        case 27: //ESC
            if(flag_menu ==0)
                close_screen();
            else if(flag_menu == 15)
                server();
            else if(flag_menu == 10)
                stMenu();
            else
                main_menu();
            break;
        default:
            break;
    }
  }
  return 0;
}
```

function.h

```
/*
        File:- function.h
        Desc:- contains all the functions related to menu control,server and client interfacing
*/
#include "var.h"
#include "sock.h"
#include "file.h"

//Server related functions
void server(){
    system("cls");
    flag_menu = 1;
    if(login_server == false){
        char pwd[20];
        char inv = '*';  //Character to be displyed for password
        char ch;
        int j=0;
        printf("\nPASSWORD PROTECTED SERVER:-\n\n");
        printf("Enter password to for server:- ");
        while(ch != 13){
            ch = getch();
            if(ch!= 13 && ch!=11 && ch!=8 ){  //13-Enter 8- Back  11- tab
                pwd[j] = ch;
                putch(inv);
                j++;
            }
        }
        pwd[j] = '\0';
        if(strcmp(PASSWORD,pwd)==0){
            login_server = true;
        }
        else{
            printf("\a\nWrong password !!!\n");
            printf("\nTry Again(y/n)");
            if(getch()=='y')
            server();
            else
            main_menu();
        }
    }
    if(login_server == true){
            system("cls");
            printf("\nSERVER OPTIONS\n\n");
            printf("1. Host a server\n");
            printf("2. Manage Student Information\n");
            printf("3. Back to main menu\n\n");
            printf("Enter a choice:-");
    }
}
//Client related functions
void client(){
    system("cls");
    printf("\nCLIENT OPTIONS\n\n");
    flag_menu = 2;
    printf("1. Join the server\n");
    printf("3. Back to main menu\n\n");
    printf("Enter a choice:-");
}

void host_server(){
    char optmsg[STRLEN];
    system("cls");
    flag_menu = 1;
    if(SOCKET_START == false){  //for the first time only
        SOCKET_START = true;
        start_socket();
        start_server();
    }
        printf("\n\t--Waiting for Client Request--\n");
        recv_data(optmsg);
        if(strcmp(optmsg,"1")==0)
            server_chat();
        else if(strcmp(optmsg,"2")==0)
            give_info();
        else
           host_server();
}

void give_info(){
    char strId[3];
    char strPercent[3];
    char strMarks[3];
    done = false;
    open_file();
    do{
        bool rec_found = false;
        recv_data(recMessage);
        printf("\nWait... Searching the required information\n");
        rewind(fp);
        while(fread(&s,recsize,1,fp) == 1){
            if(strcmp(s.roll,recMessage) == 0){
                printf("record found of %s\n",s.roll);
                send_data("FOUND");
                rec_found = true;
                printf("Sending Data....\n");
                send_data(s.regDate);
                intToString(s.id,strId);
                send_data(strId);
                send_data(s.roll);
                send_data(s.fName);
                send_data(s.lName);
                send_data(s.faculty);
                send_data(s.address);
                send_data(s.phone);
                send_data(s.email);
                for(i = 0;i<NUM_SUBJECTS;i++){
                    intToString(s.marks[i],strMarks);
                    send_data(strMarks);
                }
                intToString(s.percentage,strPercent);
                send_data(strPercent);
                //printf("END!!!\n");
                printf("Completed sending data\n\n");
                send_data("end");
            }
        }
        if(rec_found == false){
            printf("\a!!! Record Not Found\n\n");
            send_data(NOT_FOUND);
        }
        recv_data(recMessage);
    }while(strcmp(recMessage,"n")!=0);  //until user press no
    host_server();
}



void request_info(){
    do{
        done = false;
        system("cls");
        i = 0;
        int c = 0;
        printf("\n\tSEARCH THE STUDENT\n");
        printf("Enter Roll No:- ");
        gets(sendMessage);
        send_data(sendMessage);
        recv_data(recMessage);
        printf("Information Received from server:-\n");
        if( strcmp( recMessage,NOT_FOUND) == 0){
            done = true;
            printf("\a\nRecord not found !!!\a\n");
        }else{
             printf("\nRecord found !!!\n");

        while(done!=true){
            recv_data(recMessage);
            if ( strcmp( recMessage, "end" ) == 0){
                   done = true;
            }
            else{
```

```
            if(i==9)
                c = 0;
            else if(i == 15)
                c = 9;
            if(i >= 9 && i<15)
                printf("%s => %s\n",subjects[c],recMessage);
            else
                printf("%s => %s\n",request_infos[c],recMessage);
            c++;  i++;

            }
        }
        }
        printf("\nDo u want to search another record (y/n) ???");
        gets(sendMessage);
        send_data(sendMessage);
    }while(strcmp( sendMessage, "n" ) != 0);
    join_server();
}

void join_server(){
    char optmsg[STRLEN];
    flag_menu = 1;
    system("cls");
    if(SOCKET_START == false){  //for the first time only
        SOCKET_START = true;
        start_socket();
        connect_server();
    }
    do{
    system("cls");
    printf("\n\tREQUEST SERVER\n");
    printf("\nEnter the required service:-");
    printf("\n1. Chat");
    printf("\n2. Search Student Info");
    printf("\n3. Go to main");
    printf("\n\nEnter a option:-");
    gets(optmsg);
    send_data(optmsg);
    if(strcmp(optmsg,"1")==0)
        client_chat();
    else if(strcmp(optmsg,"2")==0)
        request_info();
    else if(strcmp(optmsg,"3")==0)
        main_menu();
    else
```

```
        printf("Enter the required service:- ");
    }while(1);
}
void client_chat(){
    printf("\n**The maximum character that can be sent = %d\n\n",STRLEN);
    done = false;
    do
       {
          GetAndSendMessage();
          printf("\n\t--Server is typing--\n");
          recv_data( recMessage);
          time(&t);
          printf("Recv > %s\t%s\n",recMessage,ctime(&t));
          if ( strcmp( recMessage, "end" ) == 0 ||
                     strcmp( sendMessage, "end" ) == 0 )
          {
             done = true;
          }
       }while ( !done );
    printf("\n\tPress Any key for requesting a server...");
    getch();
    join_server();
}


//host a server that listens to the clients
void server_chat(){
    printf("\n**The maximum character that can be sent = %d\n\n",STRLEN);
    done = false;
    do
       {
          printf("\n\t--Client is typing--\n");
          recv_data( recMessage);
          time(&t);
          printf("Recv > %s\t%s\n",recMessage,ctime(&t));
          GetAndSendMessage();
          if ( strcmp( recMessage, "end" ) == 0 ||
                  strcmp( sendMessage, "end" ) == 0 )
          {
              done = true;
          }
       }while ( !done );
    host_server();
}
```

```
void main_menu(){  //main menu
    system("cls");
    flag_menu = 0;
    printf("-- CLIENT SERVER APPLICATION --\n");
    printf("\nENTER THE MODE OF COMMUNICATION\n\n");
    printf("1. SERVER\n");
    printf("2. CLIENT\n");
    printf("3. Help\n");
    printf("4. Exit\n\n");
    printf("Enter a choice:-");
}

char* intToString(int n, char str[])
{
  int i = 0;              /* Loop counter              */
  int negative = 0;       /* Indicate negative integer */
  int temp = 0;           /* Temporary storage         */

  if(negative = (n<0))    /* Is it negative?  */
    n = -n;               /* make it positive */

  /* Generate digit characters in reverse order */
  do
  {
    str[i++] = '0'+n%10;    /* Create a rightmost digit        */
    n /= 10;                /* Remove the digit                */
  }while(n>0);              /* Go again if there's more digits */

  if(negative)              /* If it was negative */
    str[i++] = '-';         /* Append minus       */
  str[i] = '\0';            /* Append terminator  */

    str = strrev(str);   //reverse the string

  return str;               /* Return the string */
}
void help(){
    system("cls");
    printf("\n\n\tWelcome to the Client Server Application in C");
    printf("\n\nThis is a client server based program with two modes of operation\n");
    printf("1. Server:-\n");
    printf("==> It is password protected and server is responsible for giving services\n to
clients");
    printf("and manage information on the students so that it can be available\n across the
network\n");
    printf("\n2. Client:-\n");
    printf("==> Client can join the server by giving server ip address and port number");
    printf("\n in order to chat with server or request for  information of student.");
    gotoxy(40,20);
    printf("Press any key to continue...");
    getch();
    main_menu();
}
void welcome_screen(){
    system("cls");
    gotoxy(15,10); printf("Welcome to the Client Server Application in C");
    gotoxy(40,20); //printf("Press Any Key to Continue...");
    system("pause");

}
void close_screen(){
    system("cls");
    closesocket(Socket);
    gotoxy(20,5); printf("I/I part Mini Project on C");
    gotoxy(14,8); printf("Programmed By:-");
    gotoxy(18,10); printf("==> Ashok Basnet");
    gotoxy(18,12); printf("==> 066/BCT/505");
    gotoxy(18,14); printf("==> For more info:- http://projectsofashok.blogspot.com");
    gotoxy(40,20);// printf("Press Any Key to Continue...");
    system("pause");
    exit(0);
}
void gotoxy (int x, int y)
{
        coord.X = x; coord.Y = y; // X and Y coordinates
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

sock.h

```
/*
            File:- sock.h
            Desc:- contains the socket related functions
*/
#define DEFAULT_PORT 5000
#define STRLEN 256
int num_clients = 1;
char recMessage[STRLEN];
char sendMessage[STRLEN];

bool done = false;  //if it is true, the communication ends
```

```
SOCKET Socket;
struct client
{
    sockaddr_in c_address;
    int c_addr_len;
    SOCKET c_socket;
};
struct client c; //struct client c[n]; if multiple clients

void start_socket();
void start_server();
void connect_server();
bool recv_data(char *);
bool send_data(char *);
void GetAndSendMessage();

bool send_data(char *buffer){
    if(send( Socket, buffer, STRLEN, 0)== SOCKET_ERROR){
            done = true;

    }
    return true;
}

bool recv_data( char *buffer){
    int i = recv( Socket, buffer, STRLEN, 0 );
    buffer[i] = '\0';
   return true;
}
void GetAndSendMessage()
{
    char message[STRLEN];
    printf("Send > ");
    gets(message);
    if(strcmp(message,"end")==0)
        done = true;
    send_data( message );
}
void connect_server(){
    sockaddr_in cl_address;
    fflush(stdin);
    char ipaddress[50];
    int port;
    //char ipaddress[50] = "127.0.0.1";
    printf("\nEnter ipAddress of server:- ");
    gets(ipaddress);
    printf("Enter port number:- ");
    scanf("%d",&port);
    fflush(stdin);
    //port = DEFAULT_PORT;
    cl_address.sin_family=AF_INET;
    cl_address.sin_addr.s_addr=inet_addr(ipaddress);
    cl_address.sin_port=htons(port);
    if(connect(Socket,(sockaddr *) &cl_address,sizeof(cl_address))==SOCKET_ERROR){
        printf("\n\aError connecting to server\a\n");
        system("pause");
        WSACleanup();
         client();
    }
    else
        printf("\nConnected to server\n");
    //system("pause");
}

//start server ....bind port,,,,,and listen to incoming connections
void start_server(){
    printf("\nStarting server\n");
    sockaddr_inserver_address;
    server_address.sin_family=AF_INET;
    server_address.sin_addr.s_addr=inet_addr("0.0.0.0");
    server_address.sin_port=htons(DEFAULT_PORT);

    if(bind(Socket,(sockaddr *) &server_address,sizeof(server_address))==SOCKET_ERROR)
    {
        printf("\nFailed to bind the socket \n");
        system("pause");
        WSACleanup();
        exit(11);
    }
    else
        printf("Socket binded");

    //listening the clients
    if(listen(Socket,1)==SOCKET_ERROR)
    {
        printf("error listening on socket");
        system("pause");
        WSACleanup();
        exit(0);
    }
    else
        printf("\nAccepting Connections...");
    c.c_addr_len=sizeof(c.c_address);
    c.c_socket=accept(Socket,(sockaddr *) &(c.c_address),&c.c_addr_len);
    if(c.c_socket==SOCKET_ERROR)
        printf("couldnt accept connections");
    else{
        printf("\n\naccepted a connection");
        Socket=c.c_socket; //start a socket for the client
    }
        char *client_ip_address = inet_ntoa ( c.c_address.sin_addr );//grabs ipaddress of client
and stores that in form of string
        printf("\nclient with address is %s connected\n",client_ip_address);
        //system("pause");

}


//initialisation of windows socket
void start_socket()
{
    WSADATA wsadata;   //initiate the use of ws2_32.dll
    if(WSAStartup(MAKEWORD(2,0),&wsadata)!=0)   //MAKEWORD(2,0) makes request for
version 2.0 of winsock on system
    {
        printf("Socket Initialization: Error with WSAStartup \n required version snot availabel
\n");
        system("pause");
        WSACleanup();  //release resources
        exit(10);
    }
    //creation of socket
    Socket=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    /*
        AF_INET- specify the IPv4 address family.
        SOCK_STREAM - specify a stream socket.
        IPPROTO_TCP - specify the TCP protocol .

    */
    if (Socket==INVALID_SOCKET)
    {
        printf("Socket Initialization: Error creating socket\n");
        system("pause");
        WSACleanup();
        exit(11);
    }
    else
```

```
        printf("socket initialised");
    }
```

file.h

```
    /*
            File:- file.h
            Desc:- contains all the functions related to files and student information system
    */

    void add_student(){
        flag_menu = 10;
        system("cls");
        int tMarks = 0;
        int mark,stId;
        char strCheck[20];
        bool m_done = false;  //not entered marks correctly
        open_file();
        fseek(fp,0,SEEK_END);  //append from EOF
        printf("\n\tADD STUDENTS\n\n");
        printf("Enter the following information of a student:-\n");
        time(&t);
        strcpy(s.regDate,ctime(&t));
            printf("\nRegisterd Date:- %s \n",s.regDate);
            stId = count_records()+1;
        printf("Id: %d\n",stId);
        printf("Roll No: "); scanf("%s",strCheck);
        while(checkRoll(strCheck) == 1){
            printf("\aRoll number already exits \a !!! \n");
            printf("Enter Roll no. Again:- ");
            scanf("%s",strCheck);
        }
        s.id = stId;    strcpy(s.roll,strCheck);
        printf("First Name: "); scanf("%s",s.fName);
        printf("Last Name: "); scanf("%s",s.lName);
        printf("Address: "); scanf("%s",s.address);
            printf("E-mail: "); scanf("%s",s.email);
            printf("Phone: "); scanf("%s",s.phone);
            printf("Faculty: "); scanf("%s",s.faculty);
        printf("Marks in Exam: \n");
        for(i = 0;i<NUM_SUBJECTS;i++){
            m_done = false;
            printf("\t%s: ",subjects[i]);
            do{
                if(scanf("%d",&mark) == 1){ //its integer{
```

```
                if(mark>=0 && mark<=100){
                    s.marks[i] = mark;
                    m_done = true;
                }else{
                    printf("\a\a\tEnter mark again(0-100):-\t");
                }
            }else{
                printf("\tMark should be number \tEnter mark again(0-100):-\t");
            }

        }while(m_done != true);

        tMarks+=s.marks[i];
    }
    s.percentage = tMarks/NUM_SUBJECTS;
    //s.id = count_records()+1;
    //write to file
    fwrite(&s,recsize,1,fp);
    fflush(stdin);
    fclose(fp);

    printf("\n*** Record Added Successfully\n\n");
    printf("\n Add More Records(Y/N)? : ");
    if(getch() == 'y')
        add_student();
    else
        stMenu();
}
void view_student(){
    flag_menu = 10;
    system("cls");
    int j= count_records();
    open_file();
    rewind(fp);
    if(count_records() != 0){
        printf("\t\t\tVIEW STUDENTS\n");
        printf("\nRoll No\tName\tPhone\t\tFaculty\t\tPercentage");
        while(fread(&s,recsize,1,fp)==1){
            printf("\n%s\t",s.roll);
            printf("%s\t",s.fName);
            printf("%s\t",s.phone);
            printf("%s\t",s.faculty);
            printf("%d\t",s.percentage);
        }
    }else{
        printf("\a\n\n\tNo RECORDS FOUND !!! \a");
    }
    printf("\n\n\tTotal Students: %d\t",j);
    fflush(stdin);
    fclose(fp);
    printf("\n\nPress Esc to go back");
}

void search_student(){
    flag_menu = 10;
    char roll_no[15],another = 'y';
    int flag_rec = 0;
    open_file();
    system("cls");
    printf("\n\tSEARCH STUDENTS\n\n");
    printf("Enter roll No.: ");
    scanf("%s",roll_no);
    rewind(fp);
    while(fread(&s,recsize,1,fp) == 1){
            if(strcmp(s.roll,roll_no) == 0){
                printf("\nRegisterd Date:- %s \n",s.regDate);
                printf("\nId = %d\n",s.id);
                printf("Name = %s %s\n",s.fName,s.lName);
                printf("Roll = %s\n",s.roll);
                printf("Address = %s\n",s.address);
                printf("Phone = %s\n",s.phone);
                printf("E-mail = %s\n",s.email);
                printf("Faculty = %s\n",s.faculty);
                printf("Address = %s\n",s.address);
                printf("Marks Obtained\n");
                for(i = 0;i<NUM_SUBJECTS;i++){
                    printf("%s : \t%d \n",subjects[i],s.marks[i]);
                }
                printf("Percentage = %d\n",s.percentage);
                flag_rec = 1;
            }
    }
    if(flag_rec == 0)
        printf("\a\nNo records found\n");
    printf("\n\nSearch another student (Y/N)? ");
        fflush(stdin);
        fclose(fp);
    if(getch() == 'y')
        search_student();
    else
        stMenu();
}
```

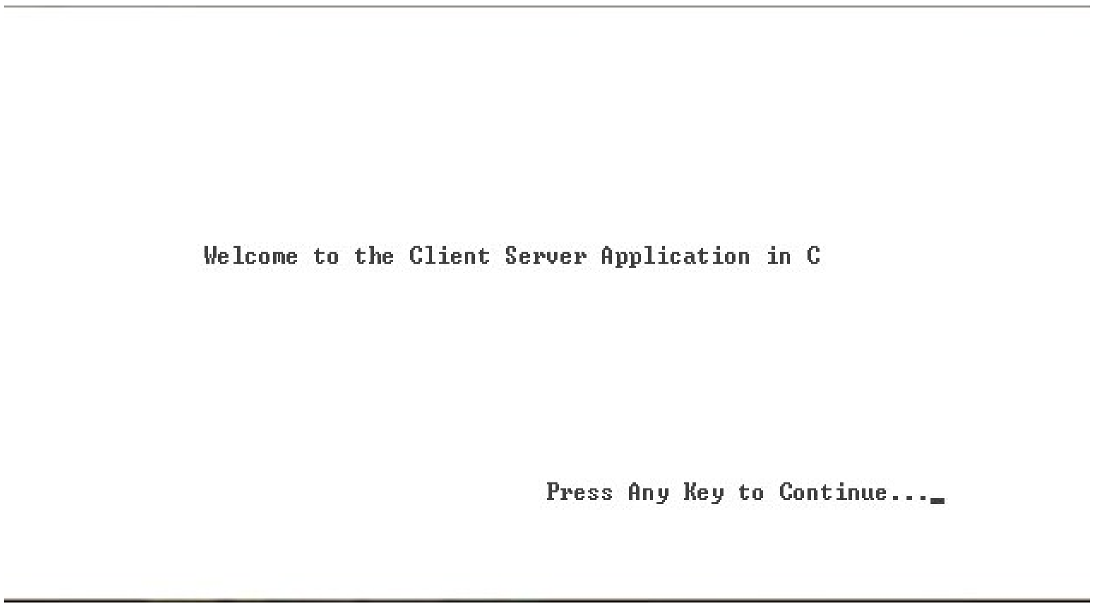
```
void edit_student(){
    bool m_done = false;  //not entered marks correctly
    int mark;
    char strEdit[20];
    flag_menu = 10;
    char roll_no[15];
    int flag_rec = 0;
    int tMarks = 0;
    open_file();
    system("cls");
    printf("\n\tEDIT STUDENTS\n\n");
    printf("Enter roll No.: ");
    scanf("%s",roll_no);
    rewind(fp);
    while(fread(&s,recsize,1,fp) == 1){
            if(strcmp(s.roll,roll_no) == 0){
                printf("\nRegisterd Date:- %s \n",s.regDate);
                printf("\nId = %d\n",s.id);
                printf("Name = %s %s\n",s.fName,s.lName);
                printf("Roll = %s\n",s.roll);
                printf("Address = %s\n",s.address);
                printf("Phone = %s\n",s.phone);
                printf("E-mail = %s\n",s.email);
                printf("Faculty = %s\n",s.faculty);
                printf("Address = %s\n",s.address);
                printf("Marks Obtained\n");
                for(i = 0;i<NUM_SUBJECTS;i++){
                    printf("%s : \t%d \n",subjects[i],s.marks[i]);
                }
                printf("Percentage = %d\n",s.percentage);
                printf("\n\nEnter the new information of a student:-\n");
                printf("\n\t !!! Enter # not to modify the current Info\n\n");
                printf("Registerd Date:- %s \n",s.regDate);
                printf("Id: %d\n",s.id);
                printf("Roll No: "); scanf("%s",strEdit);
                while(checkRoll(strEdit) == 1){
                    printf("\aRoll number already exits \a !!! \n");
                    printf("Enter Roll no. Again:- ");
                    scanf("%s",strEdit);
                }
                if(checkEmpty(strEdit)==1)
                    strcpy(s.roll,strEdit);
                printf("First Name: "); scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.fName,strEdit);
                printf("Last Name: "); scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.lName,strEdit);
                printf("Address: "); scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.address,strEdit);
                printf("E-mail: "); scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.email,strEdit);
                printf("Phone: "); scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.phone,strEdit);
                printf("Faculty: ");scanf("%s",strEdit);
                if(checkEmpty(strEdit)==1)
                    strcpy(s.faculty,strEdit);
                  printf("Marks in Exam: \n");
                for(i = 0;i<NUM_SUBJECTS;i++){
                    m_done = false;
                    printf("\t%s: ",subjects[i]);
                    do{
                        if(scanf("%d",&mark) == 1){ //its integer{
                            if(mark>=0 && mark<=100){
                                s.marks[i] = mark;
                                m_done = true;
                            }else{
                                printf("\a\a\tEnter mark again(0-100):-\t");
                            }
                        }else{
                            printf("\tMark should be number \tEnter mark again(0-100):-\t");
                        }

                    }while(m_done != true);

                    tMarks+=s.marks[i];
                }
                s.percentage = tMarks/NUM_SUBJECTS;
            //write to file
                fseek(fp,-recsize,SEEK_CUR);
                fwrite(&s,recsize,1,fp);
                flag_rec = 1;
                fclose(fp);
            }
    }
    if(flag_rec == 0)
            printf("\a\nNo records found\n");
    printf("\n\nEdit another student (Y/N)? ");
```

```
        fflush(stdin);
    if(getch() == 'y')
        search_student();
    else
        stMenu();
}
void delete_student(){
    flag_menu = 10;
    char roll_no[15];
    int flag_rec = 0;
    FILE *ftemp;
    open_file();
    system("cls");
    printf("\n\tDELETE STUDENTS\n\n");
    printf("Enter roll No.: ");
    scanf("%s",roll_no);
    ftemp = fopen("TEMP.DAT","wb");
    rewind(fp);
     while(fread(&s,recsize,1,fp) == 1){
            if(strcmp(s.roll,roll_no) == 0){
                printf("\nRegisterd Date:- %s \n",s.regDate);
                printf("Id = %d\n",s.id);
                printf("Name = %s %s\n",s.fName,s.lName);
                printf("Roll = %s\n",s.roll);
                printf("Address = %s\n",s.address);
                printf("Phone = %s\n",s.phone);
                printf("E-mail = %s\n",s.email);
                printf("Faculty = %s\n",s.faculty);
                printf("Address = %s\n",s.address);
                printf("Marks Obtained\n");
                for(i = 0;i<NUM_SUBJECTS;i++){
                    printf("%s : \t%d \n",subjects[i],s.marks[i]);
                }
                printf("Percentage = %d\n",s.percentage);
                flag_rec = 1;
            }else{
                    fwrite(&s,recsize,1,ftemp);
            }
    }
    if(flag_rec == 0)
        printf("\a\nNo records found\n");
    else{
        printf("\nAre u sure?(Y/N)");
        if(getch() == 'y'){
            fclose(fp);  fclose(ftemp);
            remove("student.DAT");
            rename("TEMP.DAT","student.DAT");
            printf("\nRecord Deleted Successfully !!!");
            remove("TEMP.DAT");

        }else{
            remove("TEMP.DAT");
            flag_rec = -1;  //no deletioni
        }
    }
    printf("\nDelete another student (Y/N)? ");
        fflush(stdin);
    if(getch() == 'y')
        delete_student();
    else
        stMenu();
}

void stMenu(){  //main menu
    system("cls");
    flag_menu = 15;
    printf("STUDENT INFORMATION SYSTEM: \n");
    printf("1. Add Student\n");
    printf("2. View Students\n");
    printf("3. Search Students\n");
    printf("4. Edit Students\n");
    printf("5. Delete Students\n");
    printf("6. Exit\n\n");
    printf("Enter a choice:-");
}


int checkEmpty(char *chkStr){
    if(strcmp(chkStr,NOTMODIFY)==0)
        return 0;
    else
        return 1;
}
int count_records(){
    int counter = 0;
    FILE *fc;
    fc = fopen("student.DAT","rb");
    fseek(fc,0,SEEK_SET);
    while(fread(&s,recsize,1,fc)==1){
            counter = s.id;
    }
    fclose(fc);
    return counter;
}
void open_file(){
    fp = fopen("student.DAT","rb+");
    if(fp == NULL){
        fp= fopen("student.DAT","wb+");
        if (fp=NULL){
            printf("Cannot open file");
            exit(1);
        }
        open_file();
    }
}

int checkRoll(char *str){
    FILE *fc;
    fc = fopen("student.DAT","rb");
    fseek(fc,0,SEEK_SET);
    while(fread(&s,recsize,1,fc)==1){
            if(strcmp(str,s.roll)==0){
                fclose(fc);
                return 1;
            }
    }
    fclose(fc);
    return 0;
}
```

Program Output

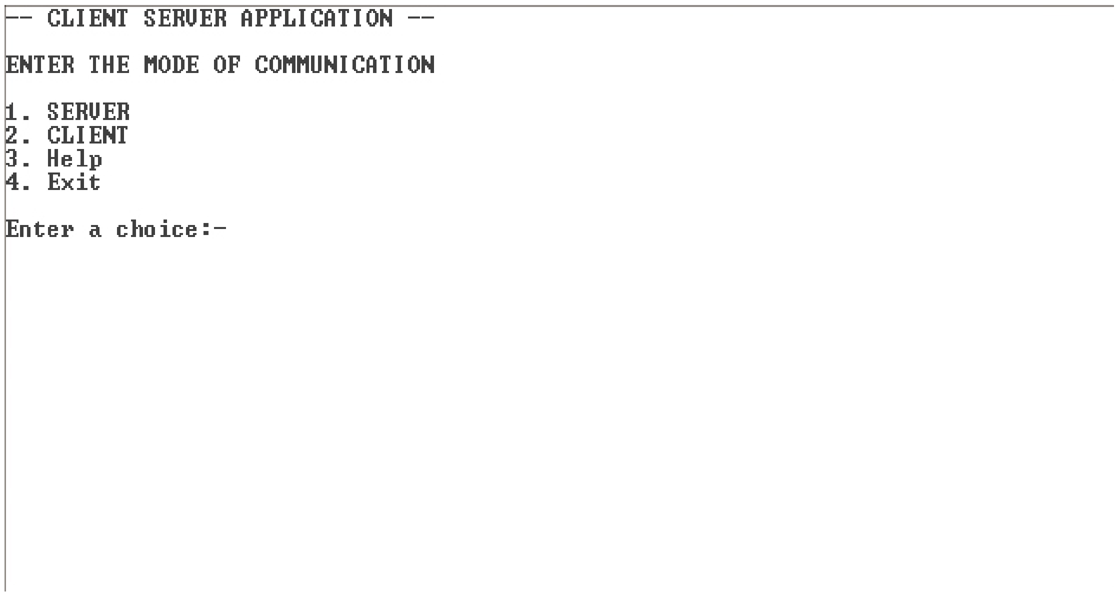
The output of the program is as given below:

1. Welcome Screen



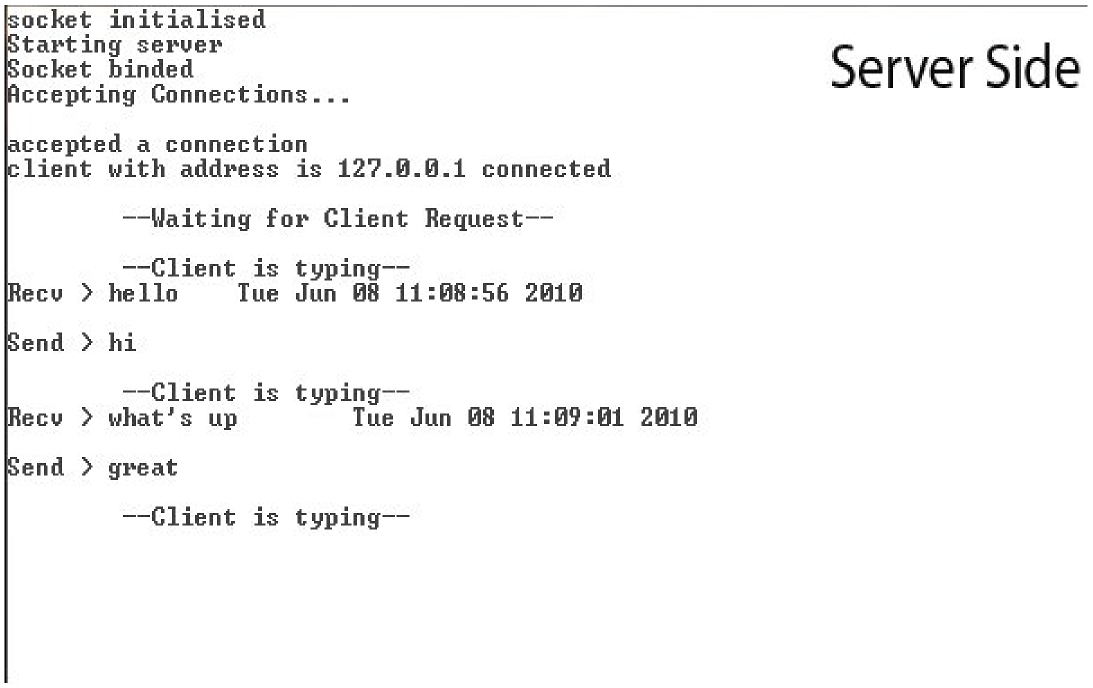
```
Welcome to the Client Server Application in C



                                 Press Any Key to Continue..._
```

2. Main interface

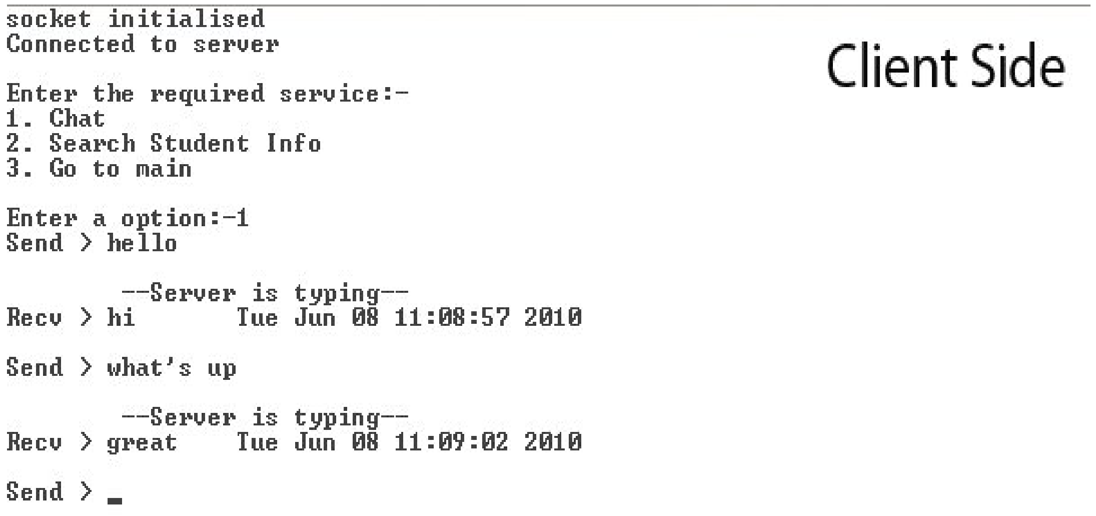


```
-- CLIENT SERVER APPLICATION --

ENTER THE MODE OF COMMUNICATION

1. SERVER
2. CLIENT
3. Help
4. Exit

Enter a choice:-
```

3. Client Serve Chat session

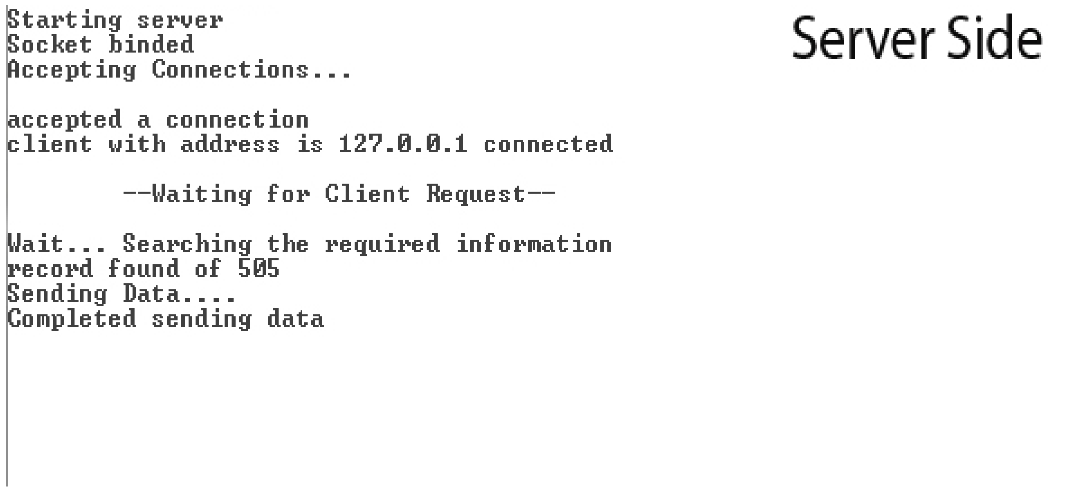


```
socket initialised
Starting server
Socket binded
Accepting Connections...

accepted a connection
client with address is 127.0.0.1 connected

        --Waiting for Client Request--

        --Client is typing--
Recv > hello    Tue Jun 08 11:08:56 2010

Send > hi

        --Client is typing--
Recv > what's up        Tue Jun 08 11:09:01 2010

Send > great

        --Client is typing--
```



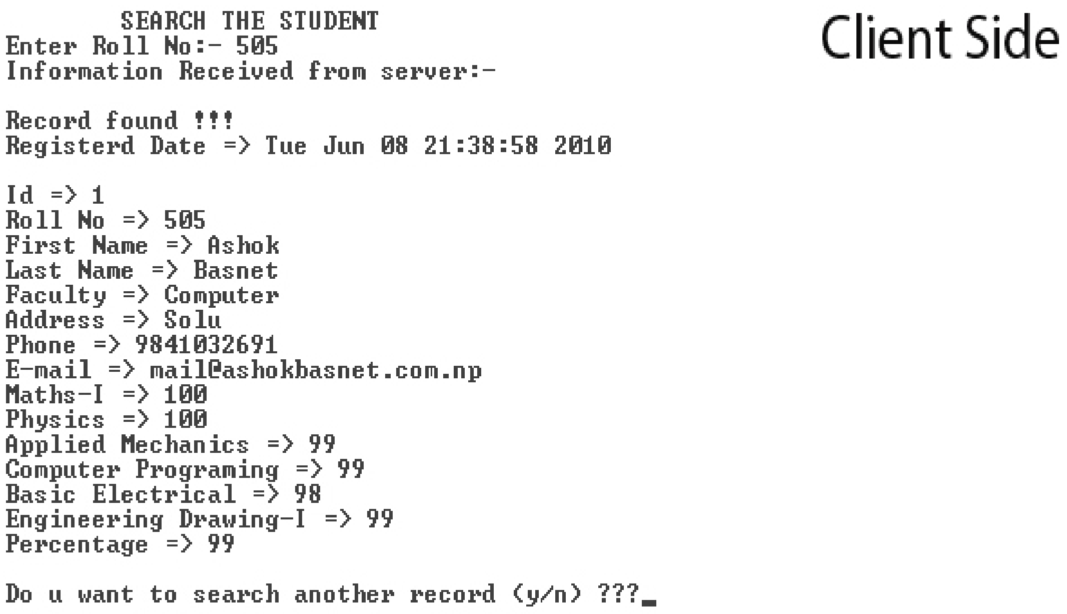
```
socket initialised
Connected to server

Enter the required service:-
1. Chat
2. Search Student Info
3. Go to main

Enter a option:-1
Send > hello

        --Server is typing--
Recv > hi       Tue Jun 08 11:08:57 2010

Send > what's up

        --Server is typing--
Recv > great    Tue Jun 08 11:09:02 2010

Send > _
```

4. Client requesting for information of a student and server replying to that



```
Starting server
Socket binded
Accepting Connections...

accepted a connection
client with address is 127.0.0.1 connected

        --Waiting for Client Request--

Wait... Searching the required information
record found of 505
Sending Data....
Completed sending data
```

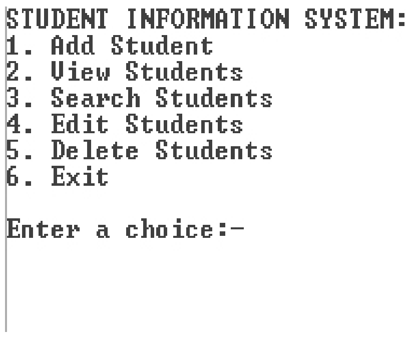
Server Side



```
        SEARCH THE STUDENT
Enter Roll No:- 505
Information Received from server:-

Record found !!!
Registerd Date => Tue Jun 08 21:38:58 2010

Id => 1
Roll No => 505
First Name => Ashok
Last Name => Basnet
Faculty => Computer
Address => Solu
Phone => 9841032691
E-mail => mail@ashokbasnet.com.np
Maths-I => 100
Physics => 100
Applied Mechanics => 99
Computer Programing => 99
Basic Electrical => 98
Engineering Drawing-I => 99
Percentage => 99

Do u want to search another record (y/n) ???_
```

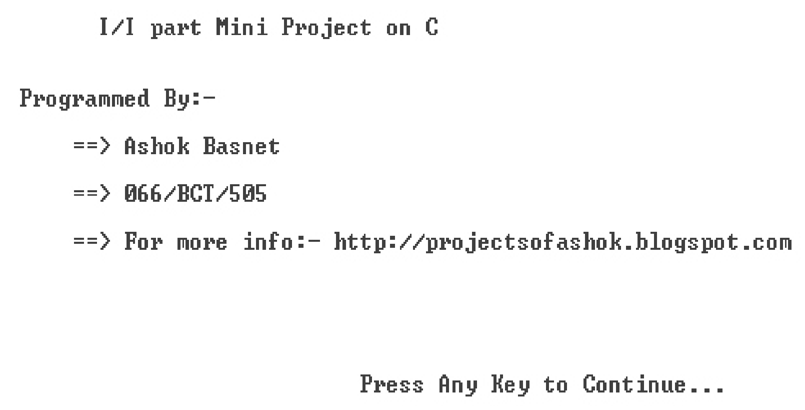
Client Side

5. Student Information System



```
STUDENT INFORMATION SYSTEM:
1. Add Student
2. View Students
3. Search Students
4. Edit Students
5. Delete Students
6. Exit

Enter a choice:-
```

6. Close Screen



```
          I/I part Mini Project on C

    Programmed By:-

        ==> Ashok Basnet

        ==> 066/BCT/505

        ==> For more info:- http://projectsofashok.blogspot.com



                              Press Any Key to Continue...
```

7. student.DAT file
   A student.DAT file is created where the information of the students are stored in binary mode. It can be manipulated by server only and client can request for the information stored in it.

Limitations

Although the program was developed to make limitations as low as possible, there are still some limitations of the programs as follows:-

1. The program runs in a blocking mode i.e. the sending and receiving of the messages is possible once at a time. One can send and receive one at time.
2. The number of client that can have communication is limited to 1.
3. Once the server has started socket, it will accept one client and bind to it until the program ends. It will just provide the requested information and processes it.

Further Enhancements

Any project can't meet its all needs once. So likewise this project can also get better enhancements to do more things and provide a better client server application in C. Following are the enhancements that can make this project more better in future.

1. The program can be made to run in non-blocking mode so that the both can send and receive data simultaneously without waiting.
2. The server can be made to accept multiple connections at a time and provide all of them with the service they need.
3. The chat system can be further taken to group chat and chatting with different computers across the network.
4. The video and voice chat can be embedded with the system so that it gives wider user interactivity.

Summary

This project was done as an application of the TCP/IP protocol and client server architecture. The project was done in a short duration of around 2 months and is successful for connecting two computers connected physically by a network, share the information between them. The application has two modes of operation, client and server. The machine running application in server mode can host the server, accept client connections and give services requested by a client. In contrary, a machine running application in client mode can join server by giving the ip address of the server and port number binded by the server for connections. There are two options for client; either he/she can chat with server to send and receive message and request for the student information that is stored and managed by the server. The student management system is a part of the server. The server is protected by a password and it has the option to either host a server or manage student information system. The basic information of the student is kept with marks and percentage. Server has the ability to edit, search, delete and view all the information regarding the information of student. Client operates in limited mode; it can search the information of the student by providing the roll number of the student to search.
The project was completed in the time frame as given by the department of Electronics and Comptuer with the great suggestions from the teachers and class mates.

References

1. Kanetkar Yashavant, Let Us C, BPB Publication, 9th Edition, 2009.

2. MSDN help site for the winsock application in Visual C++
    ⇨ http://msdn.microsoft.com/en-us/library/ms740632%28v=VS.85%29.aspx

3. Socket Programming Presentations by Vivek Ramachandran

4. www.google.com