10:44 | Posted by Muhammad Ahmed

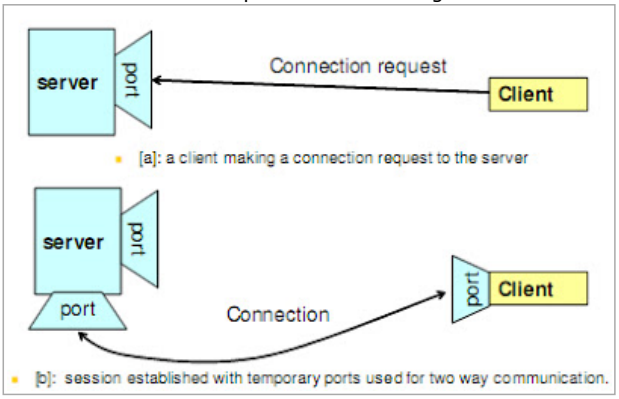
Socket Programming in C - Online Database Server

Introduction

What are Sockets? Sockets provide an interface at the transport layer. They sit in between the Transport layer and Application layer. Socket handle works in a similar way as File handle in I/O operations.

The streams used in file I/O operation are also applicable to socket-based I/O. There is a flexibility of operations. Means that a socket program based on C can works in a similar fashion in Java or VB.

A server runs on a specific computer and has a socket that is bound to a specific port. The server listens to the socket for a client to make a connection request. If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to a different port. It needs a new socket so that it can continue to listen to the original socket for connection requests while serving the connected client.



TAKEN FROM GOOGLE IMAGES

Objective

Development of an online data base server using socket programming.

Description

The program consists of two parts namely:

- **SERVER PROGRAM**
- **CLIENT PROGRAM**

The client, on the submission of a password, will gain access to the server and would then be able to get the required information.

Server program

The server contains a data base of the PNEC students. Data base includes:

- **Identification number**
- **Name**
- **Semester**
- **Discipline**

Server remains in the reception state so that any client can establish a connection with it. Upon establishment of connection between Server and Client, server asks for password form the client to login. After login client can

Labels

- .m to .exe (1)
- 4 wire resistive touch screen (2)
- 8051 Description (1)
- Alarm Gadget (1)
- Amplitude Demodulation (2)
- Amplitude Modulation (2)
- AVR (1)
- BASIC ELECTRONICS (28)
- Bit Error Rate (1)
- Block Diagrams (4)
- c6713 (2)
- calculator in matlab gui (3)
- CELLULAR SYSTEM (1)
- Communication Basics (8)
- Comparator (1)
- Computer Recycling (1)
- Counters (1)
- DAC vs PWM (1)
- Data Acquisition System (1)
- Datasheet (1)
- DB-25 (1)
- Digital Communication System (11)
- Digital Electronics (8)
- Digital Modulation (3)
- Downloads (5)
- DS1620 PROJECT (1)
- DSP (6)
- E-books (1)
- Electronics Circuits (6)
- Electronics Projects (16)
- Fan Speed Control (3)
- FDM (2)
- Gated RS Latch (2)
- GSM (1)
- GSM Projects (1)
- H-Bridge (1)
- H-Bridge with single input (1)
- How Stuff Works? (13)
- How to handle EMI problem in PCB Designing (1)
- JK Flip Flop (1)
- LCD command codes (2)
- LCD interfacing (2)
- LCD pin Description (1)
- Light Dimmer (2)
- LM324 (1)

request for information regarding any of the ids available in the database. In case the id is not available, the server notifies the client and then simply terminates the connection.

Client Program

The program for the client is written such that a client will first try to connect to the IP address of the server that the user wishes to connect. On the submission of the correct password, the client can access the database to get the required information.

Server Program C Source code (Visual Studio 2008):

```
#include<winsock2.h>
#include<stdio.h>
#include<windows.h>
#include<string.h>
#include<dos.h>

void send_data(SOCKET hClient, char id_no[6]);
void error(SOCKET hClient);
void wait();


struct info
	{
		int id_no;
		char name[15];
		char semester[5];
		char dis[15];
	};
struct info data;

int nData=0;

int main()
{

	int count=0 ;
	FILE *ptr;

	int file_size=0;
	int id=0;
	int flag=0;
	float size=0;
	int end=0;
	char buf[800] = {0};
	char  pass[15]={0};
	char pass_not[20]="password invalid";
	char pass_valid[20]="logged on";
	char id_no[6]={0};

printf( "\n\t\t================================================"
);
printf( "\n\t\t\t Welcome to student Database Server" );
printf( "\n\t\t================================================"
);


	WSADATA wsaData = {0};		// Initialize WinSock2.2 DLL
	WORD wVer = MAKEWORD(2,2);	// low word = major, highword
= minor

	int nRet = WSAStartup( wVer, &wsaData );

	if( nRet == SOCKET_ERROR )
		{
			// WSAGetLastError()
			printf( "\nFailed to init Winsock
library" );				return -1;
		}

	printf( "\n\n\n\t\tStarting server\n" );

	// name a socket

	WORD WSAEvent = 0;
	WORD WSAErr = 0;
```


Recently Commented

```
            // open a socket

            // for the server we do not want to specify a network
address
            // we should always use INADDR_ANY to allow the
protocal stack
            // to assign a local IP address

                  SOCKET hSock = {0};
                  hSock = socket( AF_INET, SOCK_STREAM,
IPPROTO_TCP );

                  if( hSock == INVALID_SOCKET )
                        {
                              printf( "\t\nInvalid socket, failed
to create socket" );
                              return -1;
                        }


                  // name socket
                  sockaddr_in saListen = {0};
                  saListen.sin_family = PF_INET;
                  saListen.sin_port = htons( 10000 );
                  saListen.sin_addr.s_addr = htonl( INADDR_ANY );


                  // bind socket's name
            nRet = bind( hSock, (sockaddr*)&saListen,
sizeof(sockaddr) );

                  if( nRet == SOCKET_ERROR )
                        {
                              printf( "\t\nFailed to bind socket"
);
                              //shutdown( hSock );

                              closesocket( hSock );
                              return -1;
                        }


      while( true )
            {
            printf( "\n\t\tListening for connections\n" );  //
listen
            nRet = listen( hSock, 5 );// connection backlog queue
set to 10

      if( nRet == SOCKET_ERROR )
            {
                  int nErr = WSAGetLastError();
                  if( nErr == WSAECONNREFUSED )
                  {
                        printf( "\nFailed to listen, connection
refused" );
                  }

      else
            {
            rintf( "\nCall to listen failed" );
            }
            closesocket( hSock );
            return -1;
      }


                              // connect

                  sockaddr_in saClient = {0};

                  int nSALen = sizeof( sockaddr );

                  SOCKET hClient={0};
                  hClient = accept( hSock, (sockaddr*)&saClient,
&nSALen );

      if( hClient == INVALID_SOCKET )
            {
                  printf( "\nInvalid client socket, connection
failed\n" );
```

Recently Added

```
                  closesocket( hSock );
                  return -1;
            }

                  printf( "\n\t\tConnection estabilished" );


again:nData=recv(hClient,pass,sizeof(pass),0);  // sending
download msg

                  if ((strcmp(pass,"NUST"))!=0&& count<5)
                        {
                              count++;

Data=send(hClient,pass_not,sizeof(pass_not),0);
                              goto again;
                        }
                  elseif ((strcmp(pass,"NUST"))==0)
                        {

nData=send(hClient,pass_valid,sizeof(pass_valid),0);
      goto down;
                        }
                  else
                        goto END;


down:       printf("\n \n\t\t:>> User logged in\n\n");

                  if((ptr=fopen("database.txt","rb+"))==NULL)
                  {
                        printf("\n File error\n");
                  }

                  nData=recv(hClient,id_no,sizeof(id_no),0);

                  id=atoi(id_no);

                  fseek(ptr,0L,SEEK_SET);

                  do
                  {
                        fread(&data,sizeof(data),1,ptr);

                        if(data.id_no==id)
                        {
                              flag=1;
                              break;
                        }


                  }while(feof(ptr)==0);

                  if(flag==1)
                  {
                  send_data(hClient,id_no);
                  flag=0;
                  nData=recv(hClient,cont,sizeof(cont),0);
                  if((strcmp(cont,"y"))==0||(strcmp(cont,"Y"))==0)
                              {
                                    goto again;
                              }
                  }
                  else
                  {
                  error(hClient);
                  }

END:
                  closesocket( hClient );             // close
client connection
                  end++;
                  if(end==10)
                        break;

                  }

                  printf( "\nShutting down the server" );

                  nRet = closesocket( hSock );    // close server
socket
```

```
                  hSock = 0;


                  if( nRet == SOCKET_ERROR )
                        {
                              printf( "\nError failed to close
socket" );
                        }
                  nRet = WSACleanup();          // Release WinSock
DLL


                  if( nRet == SOCKET_ERROR )
                        {
                              printf( "\nError cleaning up Winsock
Library" );
                              return -1;
                        }
                  printf( "\nServer is offline" );

                  return 0;                     // shut down

}


void send_data(SOCKET hClient,char id_no[6])
                  {
                        char d_found[15]="Data Found";
                        printf("\n\tSending The Record:>>");
                        printf("\n\t\t\t\t%d",data.id_no);
                        printf("\n\t\t\t\t%s",data.name);
                        printf("\n\t\t\t\t%s",data.semester);
                        printf("\n\t\t\t\t%s",data.dis);

nData=send(hClient,d_found,sizeof(d_found),0);
                        wait();
                        nData=send(hClient,id_no,sizeof(id_no),0);
                        wait();

nData=send(hClient,data.name,sizeof(data.name),0);
                        wait();

nData=send(hClient,data.semester,sizeof(data.semester),0);
                        wait();

nData=send(hClient,data.dis,sizeof(data.dis),0);
                        wait();
                  }


void error(SOCKET hClient)
            {
                        char d_error[20]="Data  Not Found";
                        wait();

nData=send(hClient,d_error,sizeof(d_error),0);
            }


void wait()
{
      unsignedint delay;
      for(delay=0;delay<=100000000;delay++)
      {
      }

}
```

Client Program C Source code (Visual Studio 2008):

```
                  #include<winsock2.h>
                  #include<ws2tcpip.h>
                  #include<stdio.h>
                  #include<string.h>
                  #include<windows.h>

                  void rec_data(SOCKET hServer);
                  int nData=0;
                  struct data
                              {
```

```
                                    char  id_no[6];
                                    char name[15];
                                    char semester[5];
                                    char dis[15];
                              };
                  struct data database;


      int main()

                        {

                              int idno=0;
                              int read=0,count=0;
                              char idno_s[5]={0};
                              char pass_con[30]={0};
                              char password[15]={0};
                              char data_check[15]={0};
                              char ipadd[15];



                  printf("please enter Ip address e.g:
172.16.64.58: ");
                  scanf("%s",ipadd);


            // Initialize WinSock2.2 DLL
            // low word = major, highword = minor

                        WSADATA wsaData = {0};
                        WORD wVer = MAKEWORD(2,2);

                        int nRet = WSAStartup( wVer, &wsaData );
                        if( nRet == SOCKET_ERROR )
                              {
                              printf( "\n Failed to init Winsock
library" );
                                    return -1;
                              }

                        printf( "\n Opening connection to server"
);
                        WORD WSAEvent = 0;
                        WORD WSAErr = 0;
                        SOCKET hServer = {0};

      // open a socket
      //
      // for the server we do not want to specify a network
address
      // we should always use INADDR_ANY to allow the protocal
stack
      // to assign a local IP address

      hServer = socket( AF_INET, SOCK_STREAM, IPPROTO_IP );
      if( hServer == INVALID_SOCKET )
            {
                  printf( "\n Invalid socket, failed to create
socket" );
                  return -1;
            }


                        // name a socket

      hostent* localHost;
      char* localIP;
      localHost = gethostbyname("");
      localIP = inet_ntoa (*(struct in_addr *)*localHost-
>h_addr_list);

                        printf("\n\n%s\n\n",localHost->h_name);
                        sockaddr_in saServer = {0};
                        saServer.sin_family = PF_INET;
                        saServer.sin_port = htons( 10000 );

                        saServer.sin_addr.s_addr
=inet_addr(ipadd);

                        // Resolve the server address and port
```

```
                        // connect

      nRet = connect( hServer, (sockaddr*)&saServer, sizeof(
sockaddr ) );
            if( nRet == SOCKET_ERROR )
                  {
                        printf( "\n Connection to server failed\n"
);
                        closesocket( hServer );
                        return -1;
                  }
printf( "\n\t\t================================================="
);
printf( "\n\t\t\t Welcome to student Database" );
printf( "\n\t\t================================================="
);

passagain:        printf("\n\n\n\tPlease enter the Password to
login :>> ");
                        scanf("%s",password);

nData=send(hServer,password,sizeof(password),0);



nData=recv(hServer,pass_con,sizeof(pass_con),0);

                        if((strcmp(pass_con,"logged on"))==0)
                                    {
                                          printf("\n\n\tLogin
Sucessfuly ");
                                          goto down;
                                    }
                        elseif (count<5)
                        {
                              goto passagain;
                        }
                        else
                              goto END;
down:       printf("\n\n\tPlease enter Id No of student to view
data: ");
                        scanf("%d",&idno);

                        sprintf(idno_s,"%d",idno);

                        nData=send(hServer,idno_s,5,0);


nData=recv(hServer,data_check,sizeof(data_check),0);

                        if((strcmp(data_check,"Data Found"))==0)
                              {

                                    printf("\n\t%s\n",data_check);
                                    rec_data(hServer);

                              }
                        else
                        printf("\n\nData Not Found\n");
END:

                        printf( "\n\tClosing connection\n" );
                        // shutdown socket
                        nRet = shutdown( hServer, SD_BOTH );
                        if( nRet == SOCKET_ERROR ) {
                        // WSAGetLastError()
                        printf( "\n Error trying to perform
shutdown on socket" );
                        return -1;
                        }
                        // close server socket
                        nRet = closesocket( hServer );
                        hServer = 0;
                        if( nRet == SOCKET_ERROR ) {
                        printf( "\n Error failed to close socket"
);
                        }
                        // Release WinSock DLL
                        nRet = WSACleanup();
                        if( nRet == SOCKET_ERROR ) {
```

```
                        printf( "\n Error cleaning up Winsock
Library" );
                        return -1;
                        }

                        return 0;
                        }



void rec_data(SOCKET hServer)
{
      nData=recv(hServer,database.id_no,sizeof(database.id_no),0);
      nData=recv(hServer,database.name,sizeof(database.name),0);

nData=recv(hServer,database.semester,sizeof(database.semester),0);
      nData=recv(hServer,database.dis,sizeof(database.dis),0);

      printf("\n\tID Number  :>> %s",database.id_no);
      printf("\n\tName       :>> %s",database.name);
      printf("\n\tSemester   :>> %s",database.semester);
      printf("\n\tDiscipline :>> %s",database.dis);

}
```

Group members: In case of any queries, feel free to contact at ***homeofgadgets@yahoo.com*** or ***elprojects@ymail.com*** or ***ahmedel619@hotmail.com***

1. Muhammad Asif
2. Ahmed Fawad <==
3. Muhammad Arslan Amin
4. Muhammad Ahmed
5. AhsanFawad
6. Kashif Ali
7. Waseem Ahmed

Posted by Muhammad Ahmed on 10:44. Filed under Electronics Projects, Socket Programming . You can follow any responses to this entry through the RSS 2.0. Feel free to leave a response

Newer Post Home Older Post