Socket Programming

王誉锡 5131309040 1558617922@163.com

1. Introduction

I implemented a client/server version and also a p2p version of file share application using **Winsock APIs.** In the client/server mode you could simply download file from any server file system. And in the p2p version you can choose one peer to connect, and in the same time you also is a server that everyone could connect you and download file from your file system.

2. Function & How to run

2.1 CLIENT/SERVER

In this version you should first execute the server application.

```
λ .\winsock_server.exe
Server is listening now
```

Now, server is listening so you could start one server application. (of course multi-client is also acceptable)

```
λ .\winsock_client.exe
Connect to where (IP_ADDRESS):
127.0.0.1
PORT:
10234
Connected!
Input the file name:
```

It will inform you to input any file name on the server side. And meantime in the server side:

```
λ .\winsock_server.exe
Server is listening now
Client connected 127.0.0.1.49687
```

It will show who connected to the server. Then you could input file name to download.

```
λ .\winsock_client.exe
Connect to where (IP_ADDRESS):
127.0.0.1
PORT:
10234
Connected!
Input the file name:
wp.png
Bytes Sent: 6
Got file from server: wp.png
Input the file name:

λ .\winsock_server.exe
Server is listening now
Client connected 127.0.0.1.49687
Bytes received: 6
Client request wp.png
File wp.png sent!
```

You will successfully get the file if the file is exist in the server side, and it will continue giving you chance to get another file until you type q.

```
λ .\winsock_client.exe
Connect to where (IP_ADDRESS):
127.0.0.1
PORT:
10234
Connected!
Input the file name:
wp.png
Bytes Sent: 6
Got file from server: wp.png
Input the file name:
q
quiting..
```

2.2 P2P

In this version every peer have same program but different port number. If you start a peer it will first create a server socket for you with the port number set in the code. And then it will act like a normal client, so you could to connect others:

```
λ .\p2pclient.exe
[Server]: Server is listening now
[Client]: Connect to where (IP_ADDRESS):

λ .\p2pclient2.exe
[Server]: Server is listening now
[Client]: Connect to where (IP_ADDRESS):
```

Input the peer's ip address that you want to connect.

```
λ .\p2pclient.exe
[Server]: Server is listening now
[Client]: Connect to where (IP_ADDRESS):
127.0.0.1
[Client]: PORT:
10235
[Client]: Connected!
[Client]: Input the file name:
[Server]: Client 127.0.0.1.47897 connected you!
wp.png
[Client]: Bytes Sent: 6
[Client]: Got file from server: wp.png
[Server]: Bytes received: 7
[Server]: Client request wp2.png
[Server]: File wp2.png sent!

λ .\p2pclient2.exe
[Server]: Server is listening now
[Client]: Connect to where (IP_ADDRESS):
127.0.0.1
[Client]: PORT:
10234
[Client]: Connected!
[Client]: Input the file name:
[Server]: Client 127.0.0.1.47129 connected you!
[Server]: Bytes received: 6
[Server]: Client request wp.png
[Server]: File wp.png sent!
wp2.png
[Client]: Bytes Sent: 7
[Client]: Got file from server: wp2.png
```

As you can see each side of peer could download file from the opposite side.

3. Analyze code

since client/server version is easier than p2p version. Thus I will only easily explain the code in the p2p version.
In the code there are three socket:
server socket to start a socket keep running as a server.
client socket to deal with the situation that some one connected you as usually.
my client socket to start a socket as a client socket just as same as client mode in the client/server version

```
int main() {
  //get ip to connect
  char IP_ADDRESS[100];
  memset(IP_ADDRESS, 0, 100);

  WSADATA WSA;
  SOCKET serverSocket, clientSocket, myclientSocket;
  struct sockaddr_in serverAddr, clientAddr, connectAddr;
  int addrLen = 0;
  int iResult = 0;
  HANDLE hThread = NULL;
```

Here is the code about how to deploy these sockets:

```
//for binding my own serverSocket
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_addr.s_addr = inet_addr(MY_ADDRESS);
  serverAddr.sin_port = htons(MY_PORT);
  memset(serverAddr.sin_zero,0x00,8);

  //init windows socket dll
  if (WSAStartup(MAKEWORD(2, 2), &WSA) != 0) {
```

```
  cout << "[Server]: Error at WSAStartup()" << endl;
  return -1;
}

//create socket
serverSocket = socket(AF_INET, SOCK_STREAM, 0);
if (serverSocket == INVALID_SOCKET) {
  cout << "[Server]: Create socket failed!" << endl;
  return -1;
}

//bind
iResult = bind(serverSocket,(struct sockaddr*)&serverAddr,sizeof(serverAddr));
if (iResult == SOCKET_ERROR) {
  cout << "[Server]: Bind failed with error" << endl;
  closesocket(serverSocket);
  WSACleanup();
  return -1;
}

//ListenSocket
if (listen(serverSocket, SOMAXCONN) == SOCKET_ERROR) {
  cout << "[Server]: Listen function failed with error" << endl;
  closesocket(serverSocket);
  WSACleanup();
  return -1;
}
cout << "[Server]: Server is listening now" << endl;


//to connect other peer
unsigned short int PORT;
cout << "[Client]: Connect to where (IP_ADDRESS): " << endl;
cin.getline(IP_ADDRESS, sizeof(IP_ADDRESS));
cout << "[Client]: PORT: " << endl;
cin >> PORT;
cin.get();
connectAddr.sin_family = AF_INET;
connectAddr.sin_addr.s_addr = inet_addr(IP_ADDRESS);
connectAddr.sin_port = htons(PORT);
memset(serverAddr.sin_zero,0x00,8);

//create myclient socket
myclientSocket = socket(AF_INET, SOCK_STREAM, 0);
if (myclientSocket == INVALID_SOCKET) {
  cout << "[Client]: Create myclientSocket failed!" << endl;
  return -1;
}

//connect
iResult = connect(myclientSocket, (struct sockaddr*) &connectAddr, sizeof(connectAddr));
if (iResult == SOCKET_ERROR) {
  cout << "[Client]: Connect failed" << endl;
  closesocket(myclientSocket);
  return -1;
}
cout << "[Client]: Connected!" << endl;
cout << "[Client]: Input the file name: " << endl;
HANDLE tThread = CreateThread(NULL, 0, MyThread, (LPVOID)myclientSocket, 0, NULL);
CloseHandle(tThread);
```

Create a socket after accepting a connection request:

```
//Accept
while(true) {
  addrLen = sizeof(clientAddr);
  clientSocket = accept(serverSocket, (struct sockaddr*) &clientAddr, &addrLen);
  if (clientSocket == INVALID_SOCKET) {
    cout << "[Server]:Accept failed with error" << endl;
    closesocket(serverSocket);
    WSACleanup();
    return -1;
  }
  cout << "[Server]: Client " << inet_ntoa(clientAddr.sin_addr) << "." << clientAddr.sin_port << " connect
  hThread = CreateThread(NULL, 0, ClientThread, (LPVOID)clientSocket, 0, NULL);
  CloseHandle(hThread);
}
```

```
  closesocket(serverSocket);
  closesocket(clientSocket);
  closesocket(myclientSocket);
  WSACleanup();
  return 0;
}
```

In the main part you can see there are two threads would be create:

```
DWORD WINAPI ClientThread(LPVOID ipParameter) {
  SOCKET clientSocket = (SOCKET)ipParameter;
  int iResult = 0;
  char buffer[BUFFER_SIZE];
  char file_name[FILE_NAME_MAX];
  // loop recive
  do {
    memset(buffer, 0, BUFFER_SIZE);
    memset(file_name, 0, FILE_NAME_MAX);
    iResult = recv(clientSocket, buffer, BUFFER_SIZE, 0);

    if (iResult > 0) {
        cout << "[Server]: Bytes received: " << iResult << endl;
        strncpy(file_name, buffer, FILE_NAME_MAX);
        cout << "[Server]: Client request " << file_name << endl;
        //open file
        FILE *f = fopen(file_name, "rb");
        if (f == NULL) {
          cout << "[Server]: File open failed" << endl;
          return -1;
        } else {
          //send data
          memset(buffer, 0, BUFFER_SIZE);
          while((iResult = fread(buffer, sizeof(char), BUFFER_SIZE, f)) > 0) {
            iResult = send(clientSocket, buffer, iResult, 0);
            if (iResult == SOCKET_ERROR) {
              cout << "[Server]: Send failed with error" << endl;
              closesocket(clientSocket);
              WSACleanup();
              return -1;
            }
            memset(buffer, 0, BUFFER_SIZE);
          }
          buffer[0] = 'e'; buffer[1] = 'o'; buffer[2] = 'f';
          iResult = send(clientSocket, buffer, 10, 0);
          fclose(f);
          cout << "[Server]: File " << file_name << " sent!" << endl;
        }
    } else if (iResult == 0) {
      cout << "[Server]: Connection closing..." << endl;
      closesocket(clientSocket);
      WSACleanup();
      return -1;
    } else {
      cout << "[Server]: Recv failed" << endl;
      closesocket(clientSocket);
      WSACleanup();
      return -1;
    }
  } while (iResult > 0);
  return 0;
}
```

This socket thread is used to deal with other client connected to us, to receive the file name and send the right file to the request. And other thread is going to execute the client part which is going to send a request file name and receive the file into local file system.

```
DWORD WINAPI MyThread(LPVOID ipParameter) {
  SOCKET myclientSocket = (SOCKET)ipParameter;
  char fileName[FILE_NAME_MAX];
  char buffer[BUFFER_SIZE];
  int iResult = 0;
  while (true) {
    memset(fileName, 0, FILE_NAME_MAX);
    cin.getline(fileName, sizeof(fileName));
    if ((fileName[0]) == 'q') {
      cout << "[Client]: quiting.." << endl;
      exit(0);
    }
    iResult = send(myclientSocket, fileName, (int) strlen(fileName), 0);
```

```
    if (iResult == SOCKET_ERROR) {
      cout << "[Client]: Send failed with error" << endl;
      closesocket(myclientSocket);
      WSACleanup();
      return -1;
    }
    cout << "[Client]: Bytes Sent: " << iResult << endl;
    FILE *f = fopen(fileName, "wb");
    if (f == NULL) {
      cout << "[Client]: can't open the a file to write" << endl;
      return -1;
    } else {
      while ((iResult = recv(myclientSocket, buffer, BUFFER_SIZE, 0)) > 0) {
        // cout << buffer << endl;
        // cout << iResult<<endl;
        if (iResult > 0 && buffer[0] == 'e' && buffer[1] == 'o' && buffer[2] == 'f')
            break;
        if (fwrite(buffer, sizeof(char), iResult, f) < iResult) {
          cout << "[Client]: Write failed" << endl;
          break;
        }
      }
      cout << "[Client]: Got file from server: " << fileName << endl;
    }
    fclose(f);
  }
}
```