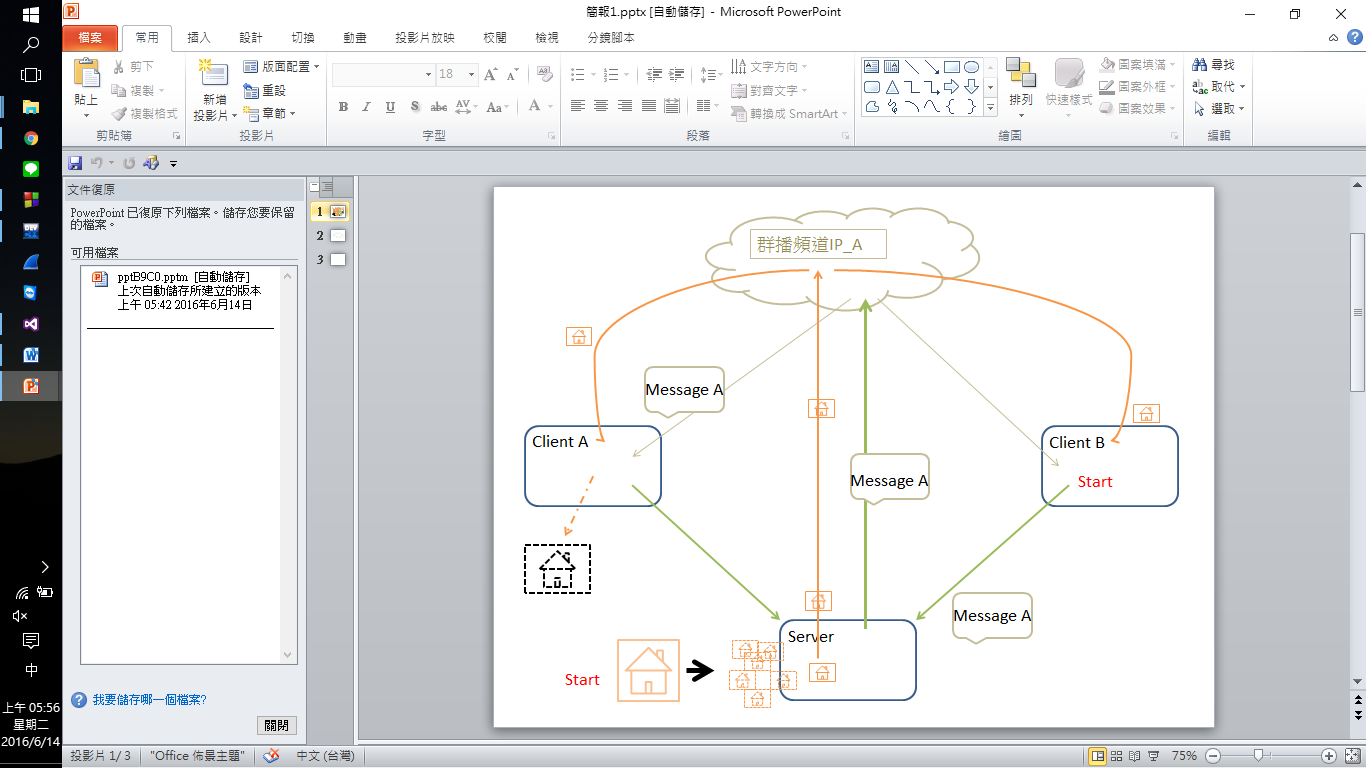
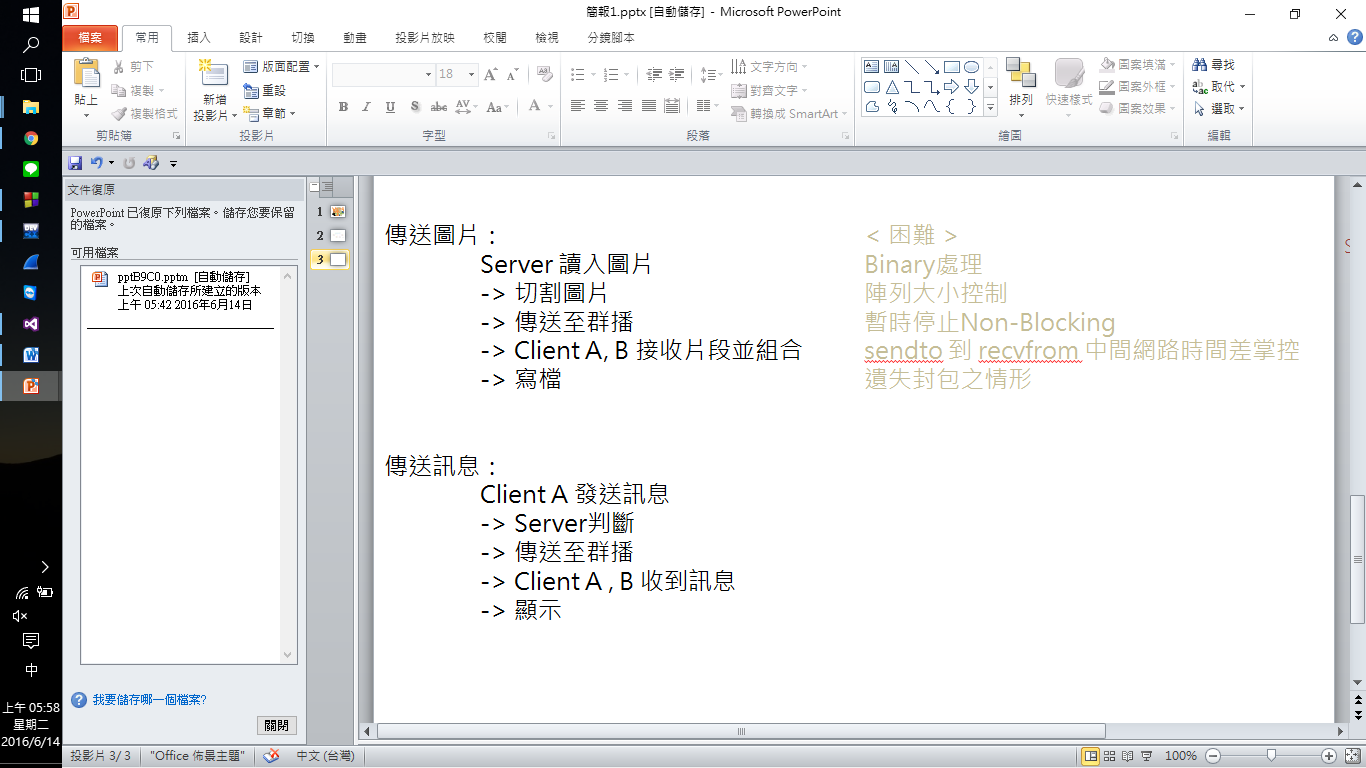
|  |
| --- |
|  |
| Winsock\_FinalProject |
|  |
| **UDP聊天室+傳送圖片** |

1. 專題摘要：
   1. 將聊天室架構精簡化
   2. 傳送圖片－不侷限於文字的傳送
2. 功能說明：
   1. 傳送圖片－打破整個學期都是傳送文字的限制
   2. Client隨時可以傳送訊息－無任何等待
   3. Server 可以完全掌握每個Client的IP位置
   4. 動態顯示Client連線情形
   5. 顯示聊天室使用者列表
   6. 無上限的連線人數
3. 程式簡述：
   1. 課程所教：
      1. 單播傳送
      2. 群播傳送
      3. Non-Blocking
   2. 自行設計：
      1. 讀入圖片以Binary型態去處理
      2. 將大圖片切割成數個小封包串流傳送
      3. 將數個小圖片封包組合成完整圖片檔案
      4. kbhit() －動態抓取使用者鍵盤輸入情形

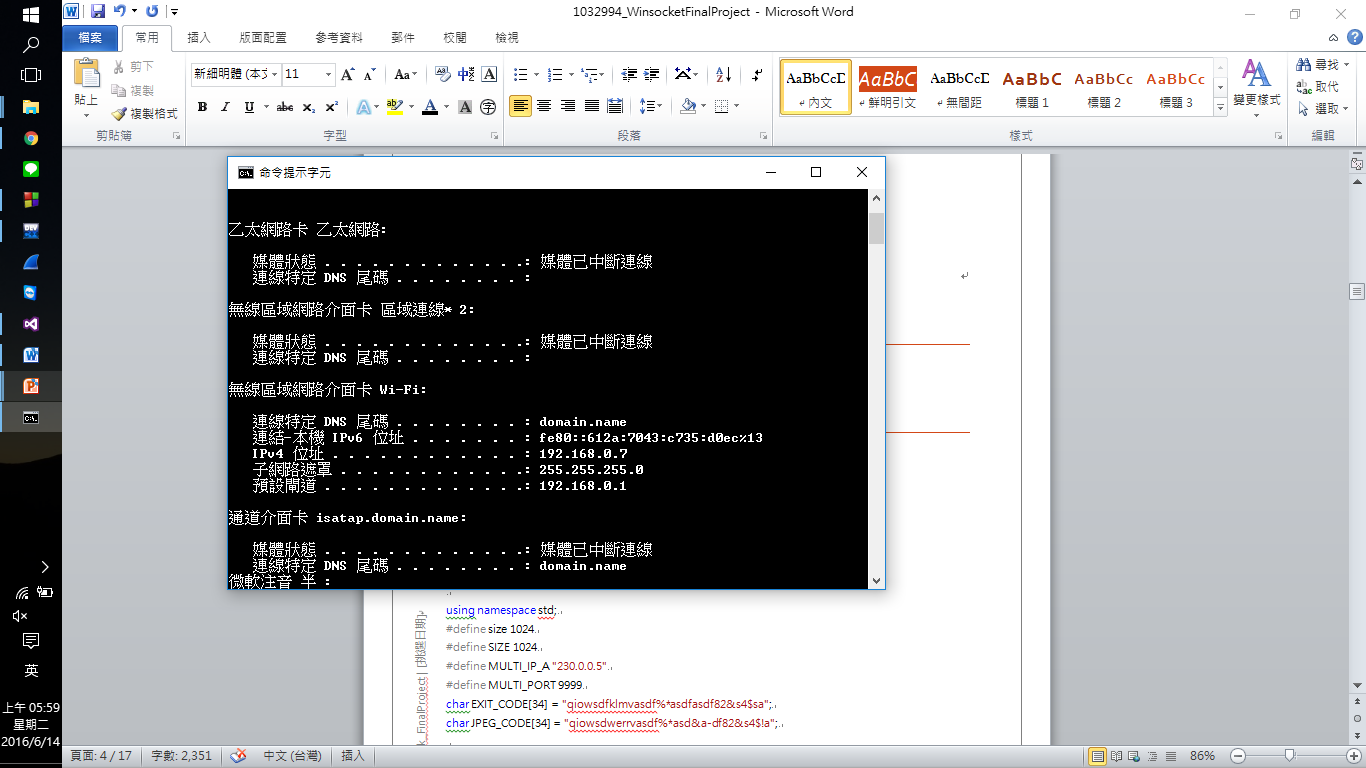
**程式架構圖**：



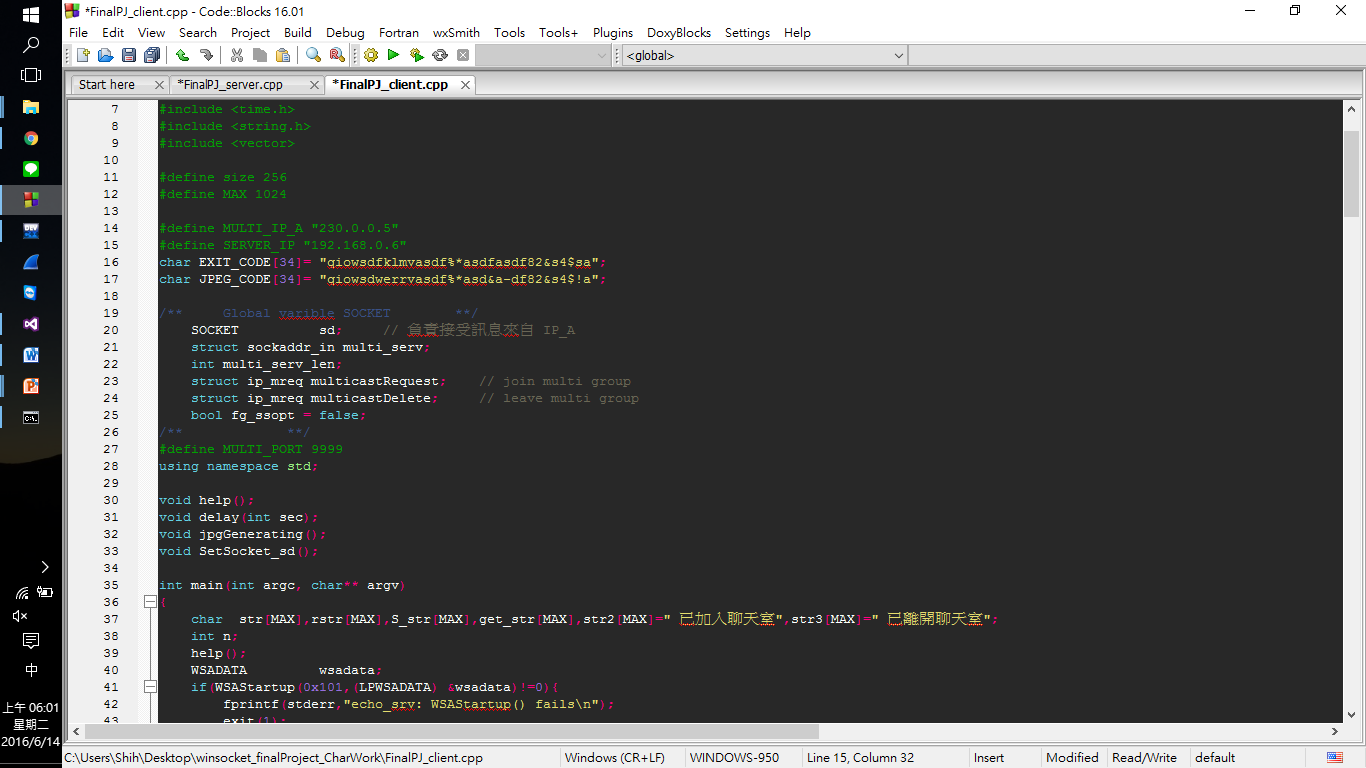
**程式遭遇的困難**：



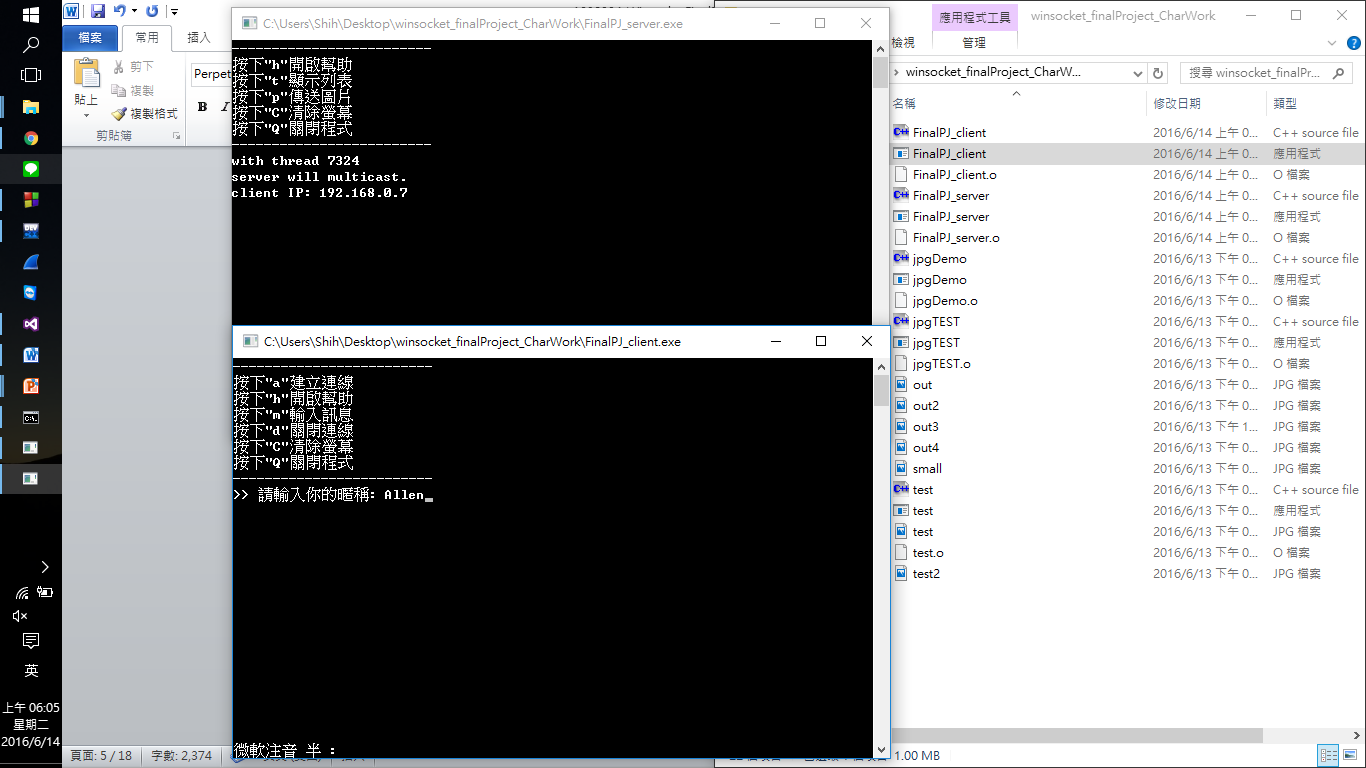
1. 執行畫面：
   1. 取得Server IP位置



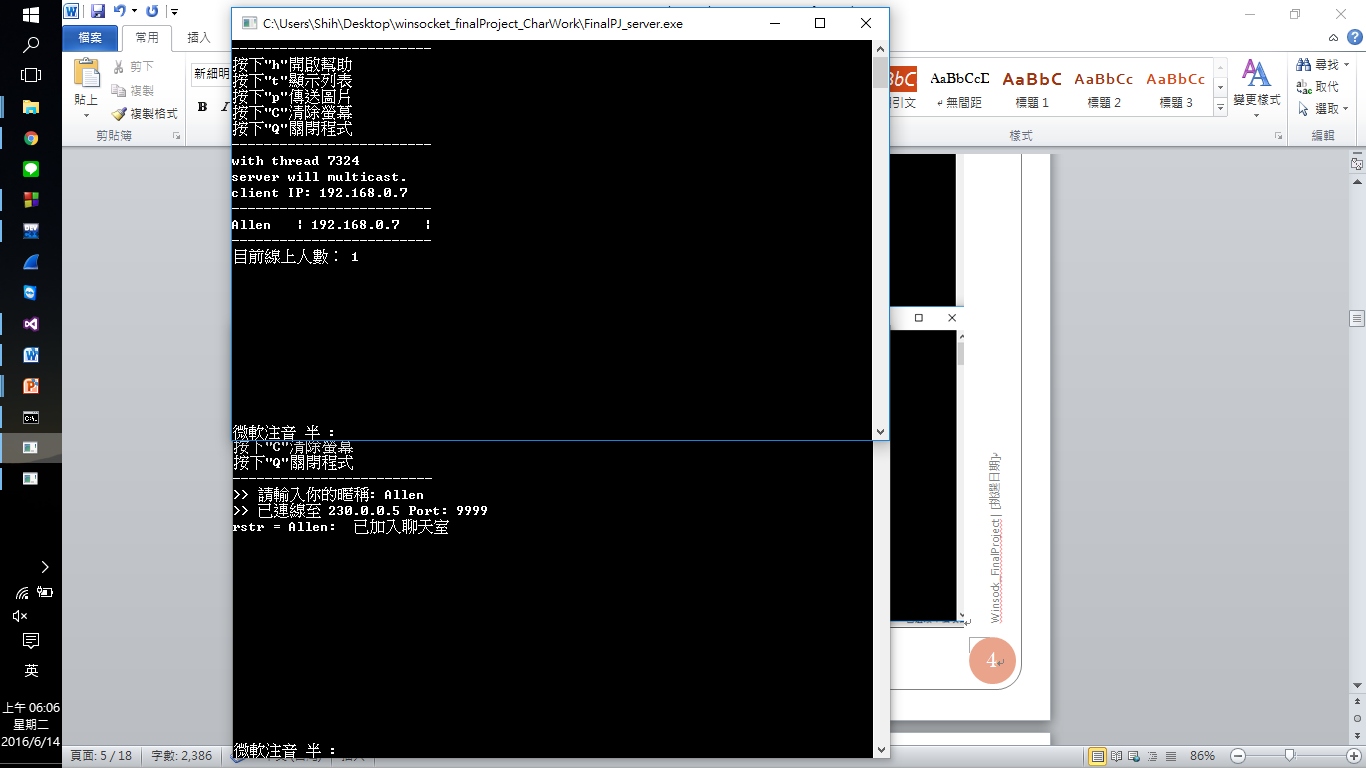
* 1. 更改Client .cpp #define SERVER\_IP 的部分



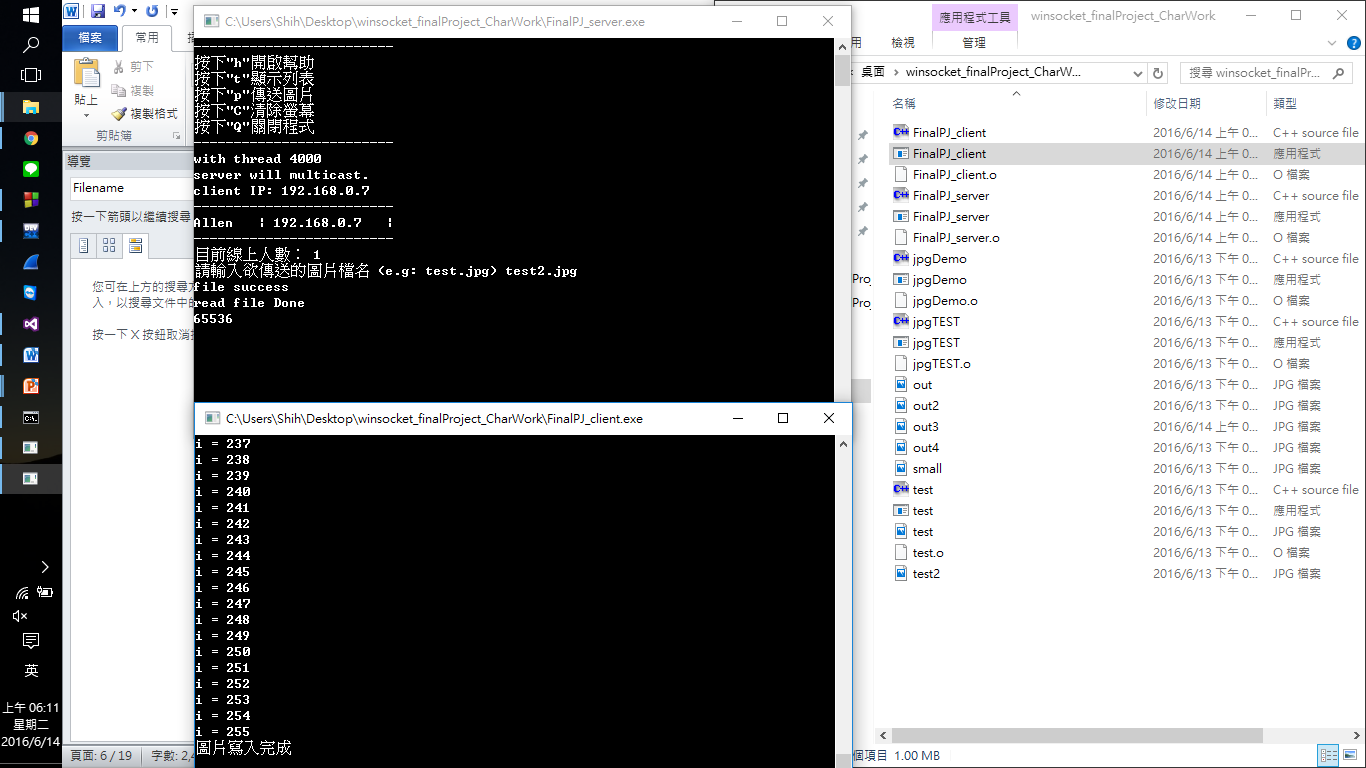
* 1. 執行Server後執行 Client，Client 按下’a’ 輸入暱稱



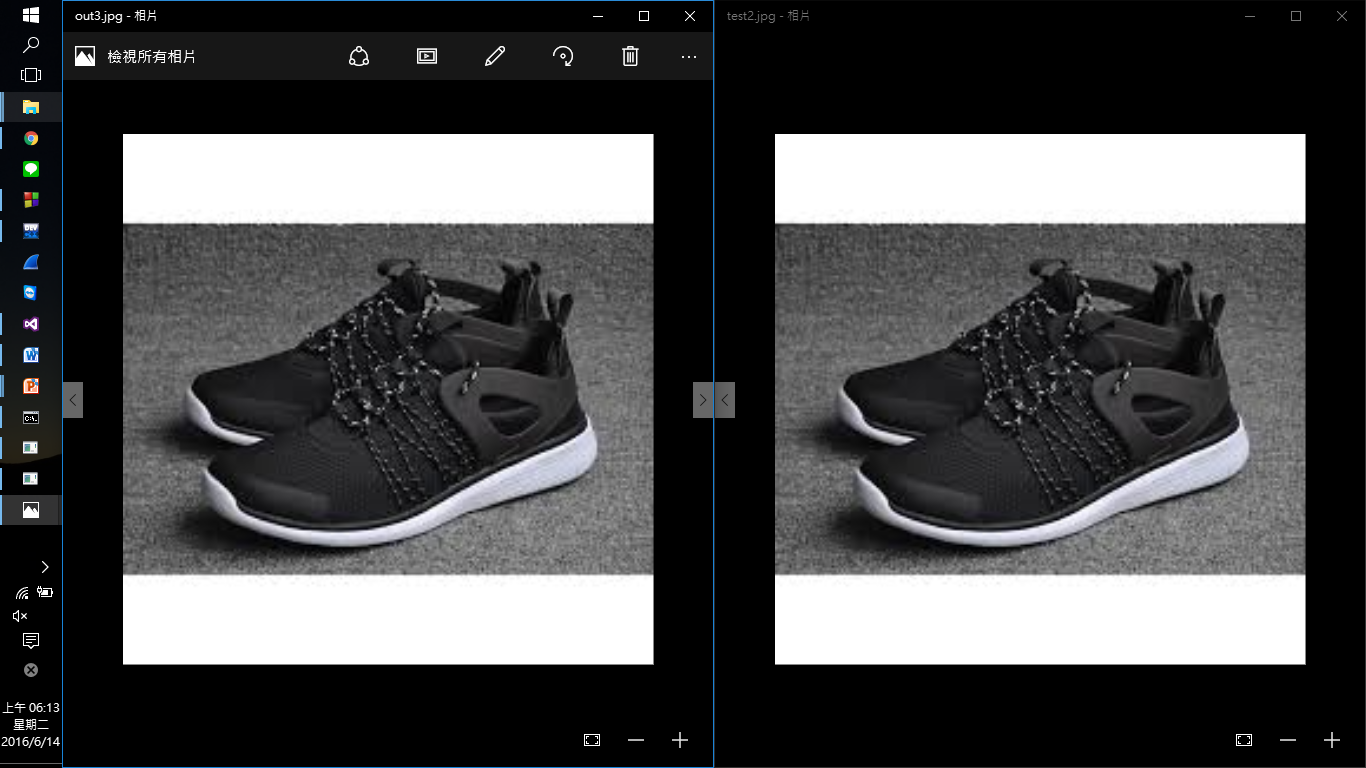
* 1. 按下t可以顯示連入的使用者列表



* 1. 使用第二台電腦並且與server連在同一個區網底下，開啟client並執行以上步驟
  2. Server按下p產生傳送圖片視窗



* 1. 檢查輸出的檔案”out3.jpg” 是否與原輸入檔案相同



* 1. 同樣的步驟可以在第二台Client電腦看到相同的結果，詳請參閱DEMO介紹

https://youtu.be/p1iPBHbwnwY

1. 程式碼：

/\* server \*/

#include <iostream>

#include <winsock.h>

#include <stdio.h>

#include <conio.h>

#include <vector>

using namespace std;

#define size 1024

#define SIZE 1024

#define MULTI\_IP\_A "230.0.0.5"

#define MULTI\_PORT 9999

char EXIT\_CODE[34] = "qiowsdfklmvasdf%\*asdfasdf82&s4$sa";

char JPEG\_CODE[34] = "qiowsdwerrvasdf%\*asd&a-df82&s4$!a";

void \*ThreadMain(void \*arg); /\* Main program of a thread \*/

void help();

void printClntTable();

void jpgSending(int serv\_sd, sockaddr\_in serv);

string clntTable[SIZE][2];

int tableSize = 0;

struct ThreadArgs /\* Structure of arguments to pass to client thread \*/

{

int clntIndex;

int clntSock; /\* Socket descriptor for client \*/

sockaddr\_in clntAddr;

};

int main()

{

help();

struct ThreadArgs \*threadArgs; /\* Pointer to argument structure for thread \*/

DWORD threadID; /\* Thread ID from CreateThread() \*/

int clntIndex = -1; /\* Index for client table \*/

int servSock; /\* Socket descriptor for server \*/

int clntSock; /\* Socket descriptor for client \*/

WSADATA wsaData; /\* Structure for WinSock setup communication \*/

struct sockaddr\_in echoServAddr; /\* Local address \*/

struct sockaddr\_in echoClntAddr; /\* Client address \*/

int clntLen; /\* Length of client address data structure \*/

if (WSAStartup(MAKEWORD(2, 0), &wsaData) != 0) /\* Load Winsock 2.0 DLL \*/

{

printf("WSAStartup() failed");

exit(1);

}

/\* Create socket for incoming connections \*/

if ((servSock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

printf("socket() failed");

/\* Construct local address structure \*/

echoServAddr.sin\_family = AF\_INET; /\* Internet address family \*/

echoServAddr.sin\_addr.s\_addr = 0; /\* Any incoming interface \*/

echoServAddr.sin\_port = htons(7788); /\* Local port \*/

/\* Bind to the local address \*/

if (bind(servSock, (LPSOCKADDR)&echoServAddr, sizeof(echoServAddr)) < 0)

printf("bind() failed");

/\* Mark the socket so it will listen for incoming connections \*/

if (listen(servSock, 10) < 0)

printf("listen() failed");

for (;;) /\* Run forever \*/

{

clntLen = sizeof(echoClntAddr);

/\* Wait for a client to connect \*/

if ((clntSock = accept(servSock, (LPSOCKADDR)&echoClntAddr, &clntLen)) < 0)

printf("accept() failed");

clntIndex++;

tableSize++;

/\* Create separate memory for client argument \*/

threadArgs = (struct ThreadArgs \*) malloc(sizeof(struct ThreadArgs));

threadArgs->clntSock = clntSock;

threadArgs->clntAddr = echoClntAddr;

threadArgs->clntIndex = clntIndex;

if (CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ThreadMain, threadArgs,

0, (LPDWORD)&threadID) == NULL)

printf("thread\_create() failed");

printf("with thread %ld\n", threadID);

}

}

void \*ThreadMain(void \*threadArgs)

{

char str[2] = "", rstr[SIZE];

int n;

bool flag = false; // flag for get Client name

sockaddr\_in clntAddr = ((struct ThreadArgs \*) threadArgs)->clntAddr; /\*\* Get client info \*\*/

int clntSock = ((struct ThreadArgs \*) threadArgs)->clntSock; /\* Socket descriptor for client connection \*/

int clntIndex = ((struct ThreadArgs \*) threadArgs)->clntIndex;

free(threadArgs); /\* Deallocate memory for argument \*/

int len\_clntAddr = sizeof(clntAddr);

/\*\* let socket-client become non-blocking \*\*/

u\_long iMode = 1;

ioctlsocket(clntSock, FIONBIO, &iMode);

/\*\* Create a multi-socket \*\*/

SOCKET serv\_sd = socket(AF\_INET, SOCK\_DGRAM, 0);

sockaddr\_in serv;

int multicastTTL = 1;

if (setsockopt(serv\_sd, IPPROTO\_IP, IP\_MULTICAST\_TTL,

(char \*)&multicastTTL, sizeof(multicastTTL)) == SOCKET\_ERROR)

cout << "setsockopt() failed" << endl;

cout << "server will multicast." << endl;

int serv\_len = sizeof(serv);

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(MULTI\_PORT);

serv.sin\_addr.s\_addr = inet\_addr(MULTI\_IP\_A);

/\*\* ------------------------------------------------------------- \*/

/\*\* Build Client Table \*\*/

getpeername(clntSock, (LPSOCKADDR)&clntAddr, &len\_clntAddr);

printf("client IP: %s\n", inet\_ntoa(clntAddr.sin\_addr));

clntTable[clntIndex][1] = inet\_ntoa(clntAddr.sin\_addr); // Setting client IP in Client Table

/\*\* Setting client name in Client Table \*\*/

while (1) {

n = recv(clntSock, rstr, SIZE, 0); rstr[n] = '\0';

if (n > 0) {

clntTable[clntIndex][0] = rstr;

break;

}

}

for (;;)

{

n = recv(clntSock, rstr, SIZE, 0); // mono-cast getting message from client

//rstr[n] = '\0';

if (n > 0) { /\*\* have get message from client \*\*/

if (strcmp(rstr, EXIT\_CODE) == 0) {

cout << "in!" << endl;

swap(clntTable[clntIndex][0], clntTable[tableSize - 1][0]);

swap(clntTable[clntIndex][1], clntTable[tableSize - 1][1]);

if (tableSize >0) tableSize -= 1;

}

else

sendto(serv\_sd, rstr, strlen(rstr) + 1, 0, (LPSOCKADDR)&serv, serv\_len); // forwarding the message to multi-IP

}

if (\_kbhit()) {

char selection;

selection = \_getch();

switch (selection) {

case'p': //傳送圖片

sendto(serv\_sd, JPEG\_CODE, strlen(JPEG\_CODE) + 1, 0, (LPSOCKADDR)&serv, serv\_len); // 傳送認證碼給Client判斷

jpgSending(serv\_sd, serv);

break;

case't': // 印出目前連線列表及線上人數

printClntTable();

break;

case'h': // 印出小幫手

help();

break;

case'C': //清除螢幕

system("cls");

help();

break;

case'Q': //關閉程式

WSACleanup();

exit(1);

break;

}

}

}

}

void help()

{

cout << "-------------------------" << endl;

printf("按下\"h\"開啟幫助\n");

printf("按下\"t\"顯示列表\n");

printf("按下\"p\"傳送圖片\n");

printf("按下\"C\"清除螢幕\n");

printf("按下\"Q\"關閉程式\n");

cout << "-------------------------" << endl;

}

void printClntTable()

{

cout << "-------------------------" << endl;

for (int i = 0; i<tableSize; i++) {

for (int j = 0;j<2;j++) {

cout << clntTable[i][j] << "\t| ";

}

cout << endl << "-------------------------" << endl;

}

cout << "目前線上人數： " << tableSize << endl;

}

void jpgSending(int serv\_sd, sockaddr\_in serv)

{

vector<char> vc;

int serv\_len = sizeof(serv);

unsigned char image\_buf[size\*size + 1];

FILE \*Fimage;

string Filename ;

cout << "請輸入欲傳送的圖片檔名 (e.g: test.jpg) ";

cin >> Filename;

Fimage = fopen(Filename.c\_str(), "rb");

if (!Fimage) {

cout << "FILE Error! " << endl;

}

else {

cout << "file success" << endl;

fread(&image\_buf, 1, size\*size, Fimage);

}

cout << "read file Done" << endl;

fclose(Fimage);

char sendbuf[size] = "";

int count = 0;

for (int i = 0;i<size;i++) {

for (int j = 0;j<size;j++) {

sendbuf[j] = image\_buf[size\*i + j];

count++;

}

sendto(serv\_sd, sendbuf, size, 0, (LPSOCKADDR)&serv, serv\_len);

Sleep(3);

}

cout << count << endl;

//sendto(serv\_sd,image\_buf,size\*size,0,(LPSOCKADDR)&serv,serv\_len);

}

/\* client.cpp \*/

#include <stdio.h>

#include <string.h>

#include <winsock.h>

#include <iostream>

#include <conio.h>

#include <time.h>

#include <string.h>

#include <vector>

#define size 256

#define MAX 1024

#define MULTI\_IP\_A "230.0.0.5"

#define SERVER\_IP "192.168.0.6"

char EXIT\_CODE[34] = "qiowsdfklmvasdf%\*asdfasdf82&s4$sa";

char JPEG\_CODE[34] = "qiowsdwerrvasdf%\*asd&a-df82&s4$!a";

/\*\* Global varible SOCKET \*\*/

SOCKET sd; // 負責接受訊息來自 IP\_A

struct sockaddr\_in multi\_serv;

int multi\_serv\_len;

struct ip\_mreq multicastRequest; // join multi group

struct ip\_mreq multicastDelete; // leave multi group

bool fg\_ssopt = false;

/\*\* \*\*/

#define MULTI\_PORT 9999

using namespace std;

void help();

void delay(int sec);

void jpgGenerating();

void SetSocket\_sd();

int main(int argc, char\*\* argv)

{

char str[MAX], rstr[MAX], S\_str[MAX], get\_str[MAX], str2[MAX] = " 已加入聊天室", str3[MAX] = " 已離開聊天室";

int n;

help();

WSADATA wsadata;

if (WSAStartup(0x101, (LPWSADATA)&wsadata) != 0) {

fprintf(stderr, "echo\_srv: WSAStartup() fails\n");

exit(1);

}

// --------------------------------------------------------//

/\*\* Create socket to mono-send message to server \*\*/

SOCKET serv\_sd = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in serv;

int serv\_len = sizeof(serv);

serv.sin\_family = AF\_INET;

serv.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);

serv.sin\_port = htons(7788);

connect(serv\_sd, (SOCKADDR\*)&serv, sizeof(serv));

// -----------------------------------------------------

/\*\* Setting the socket sd \*\*/

SetSocket\_sd();

// -------------------------------------------------------/

/\*\* Start using non-blocking \*\*/

//enable non-blcking mode when iMode=1

u\_long iMode = 1;

ioctlsocket(sd, FIONBIO, &iMode); /\*\* let socket-sd become non-blocking \*\*/

u\_long iMode2 = 1;

ioctlsocket(serv\_sd, FIONBIO, &iMode2);

bool connect = false; // 判斷是否已經連線

char name[MAX]; // 儲存Client 的暱稱

int Nlength = 0; // Length of Client's name

bool flag = false; // 判斷是否進入傳送圖片模式

int k;

char image\_buf[size + 1];

unsigned char image[size\*size + 1];

FILE \*Fresult;

int count = 0;

while (1)

{

if (connect) {

n = recvfrom(sd, rstr, MAX, 0, (LPSOCKADDR)&multi\_serv, &multi\_serv\_len); // receive message from multi-IP

rstr[n] = '\0';

if (n >= 0) {

if (strcmp(rstr, JPEG\_CODE) == 0) // Server 要開啟傳送模式必須先傳送一個特定的字串，在此做比較

flag = true;

if (flag == true) {

/\*\* 關閉 socket sd 並重新設定 (目的：關閉non-blocking Mode) \*\*/

closesocket(sd);

SetSocket\_sd();

Fresult = fopen("out3.jpg", "wb");

//cout << "圖片產生中...." << endl;

for (int i = 0;i<size;i++) {

k = recvfrom(sd, image\_buf, size, 0, (LPSOCKADDR)&multi\_serv, &multi\_serv\_len);

cout << "i = " << i << endl;

//cout << k << endl;

for (int j = 0;j<size;j++) {

count++;

image[i\*size + j] = image\_buf[j];

}

}

fwrite(&image, 1, size\*size, Fresult);

cout << "圖片寫入完成" << endl;

fclose(Fresult);

flag = false;

/\*\* 開啟 non-blocking \*\*/

iMode = 1;

ioctlsocket(sd, FIONBIO, &iMode);

}

else

cout << "rstr = " << rstr << endl;

}

}

if (\_kbhit()) {

char selection;

selection = \_getch();

switch (selection) {

case'a': // 開始連線

cout << ">> 請輸入你的暱稱: ";

cin >> name; cin.get();

send(serv\_sd, name, strlen(name) + 1, 0); //傳送姓名資訊給server

Nlength = strlen(name);

name[Nlength] = ':'; name[Nlength + 1] = ' '; name[Nlength + 2] = '\0';

connect = true;

cout << ">> 已連線至 " << MULTI\_IP\_A << " Port: " << MULTI\_PORT << endl;

/\*\* 群播: Client A 已上線 \*\*/

strcpy(S\_str, name);

strcat(S\_str, str2);

send(serv\_sd, S\_str, strlen(S\_str) + 1, 0);

break;

case'm': // 讓client輸入訊息

if (connect) {

cout << ">> 請輸入訊息: ";

cin.getline(get\_str, MAX, '\n');

strcpy(S\_str, name);

strcat(S\_str, get\_str);

send(serv\_sd, S\_str, strlen(S\_str) + 1, 0); // mono-send to the server

}

else cout << ">> 連線遺失，請按'a'來重新建立連線 " << endl;

break;

case'd': // 刪除連線

multicastDelete.imr\_multiaddr.s\_addr = inet\_addr(MULTI\_IP\_A);

multicastDelete.imr\_interface.s\_addr = htonl(INADDR\_ANY); // INADDR\_ANY 為預設網卡(大部分電腦只有一張網卡)

if (setsockopt(sd, IPPROTO\_IP, IP\_DROP\_MEMBERSHIP,

(char \*)&multicastDelete, sizeof(multicastDelete)) < 0) {

cout << ">> 已經沒有連線可以刪除了耶..." << endl;

break;

}

cout << ">> 正在刪除連線..."; delay(3); cout << "OK!" << endl;

connect = false;

strcpy(S\_str, name);

strcat(S\_str, str3);

send(serv\_sd, S\_str, strlen(S\_str) + 1, 0); // 離線訊息

send(serv\_sd, EXIT\_CODE, strlen(EXIT\_CODE) + 1, 0);// 傳送離線代碼，為使Server將此

name[0] = '\0';

system("pause");

exit(1);

break;

case'h': // 印出小幫手

help();

break;

case'C': //清除螢幕

system("cls");

help();

break;

case'Q': //關閉程式

if (name[0] == '\0') name[0] = '?';

strcpy(S\_str, name);

strcat(S\_str, str3);

send(serv\_sd, S\_str, strlen(S\_str) + 1, 0); // 離線訊息

send(serv\_sd, EXIT\_CODE, strlen(EXIT\_CODE) + 1, 0);// 離線代碼

delay(2);

closesocket(sd);

WSACleanup();

exit(1);

break;

}

}

}

send(serv\_sd, EXIT\_CODE, strlen(EXIT\_CODE) + 1, 0);// 離線代碼

closesocket(sd);

closesocket(serv\_sd);

WSACleanup();

return 0;

}

void help()

{

cout << "-------------------------" << endl;

printf("按下\"a\"建立連線\n");

printf("按下\"h\"開啟幫助\n");

printf("按下\"m\"輸入訊息\n");

printf("按下\"d\"關閉連線\n");

printf("按下\"C\"清除螢幕\n");

printf("按下\"Q\"關閉程式\n");

cout << "-------------------------" << endl;

}

void delay(int sec)

{

time\_t \_t;

\_t = time(NULL);

while (time(NULL) <= \_t + sec - 1) {};

}

void jpgGenerating()

{

}

void SetSocket\_sd()

{

/\*\* Create socket receive from multi group \*\*/

sd = socket(AF\_INET, SOCK\_DGRAM, 0);

multi\_serv\_len = sizeof(multi\_serv);

multi\_serv.sin\_family = AF\_INET;

multi\_serv.sin\_addr.s\_addr = 0;

multi\_serv.sin\_port = htons(MULTI\_PORT);

if (bind(sd, (LPSOCKADDR)&multi\_serv, sizeof(multi\_serv))<0) {

printf("bind error!\n");

}

multicastRequest.imr\_multiaddr.s\_addr = inet\_addr(MULTI\_IP\_A); // specify the multi group

multicastRequest.imr\_interface.s\_addr = htonl(INADDR\_ANY); // 選擇要使用的網卡

if (setsockopt(sd, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP,

(char \*)&multicastRequest, sizeof(multicastRequest)) < 0)

cout << "socket\_sd setsockopt error" << endl;

else

fg\_ssopt = true;

}

1. 感想/心得/未來期待：

在完成這次專題之後，發現自己對於單播、群播，不管是語法還是概念都比之前還要更上一層樓的感覺。

很享受實做出自己想要功能的感覺，雖然在處理傳送圖片的過程中遇到了許多困難，翻偏了中外各種文章文件，曾經一度已經想要放棄，很感謝自己最後有堅持並且時做出來。

未來期待：不限於圖片大小,均能傳送、傳送ppt 等其他檔案格式。