張維倫

資訊三丙 D0672458

作業系統HW2

Sockets、Multithreaded Program

**目錄**

**目錄 -- 編頁碼-**-----------------------------------------------------------------------------1

**題目定義 -- 作法及說明-**---------------------------------------------------------2

**程式碼**------------------------------------------------------------------------------------------------3-4

**執行結果 -- 測試資料及結果**----------------------------------------------------5

**討論 -- 遭遇困難及解決過程-**---------------------------------------------------6

**心得 -- 自由發揮**---------------------------------------------------------------------------7

**題目定義 -- 作法及說明**

**程式說明：**

* 1. 結合Sockets與Multithreading的概念，分別撰寫server.cpp與client.cpp兩支程式，完成聊天室。
     1. 必須允許多個client端向一個server端進行連線。
     2. 當有一方client方發送訊息時，其餘所有client端都應收到訊息。

**作法 :**

從Sockets到連接伺服器的步驟和執行緒的架構大致上都已經完成了，所以我從PPT上把所有的functions都瞭解了一下，然後再看看自己要往哪裡下手，大概知道所有的函式功能之後，我便先從Server的執行緒進行改建工程，再從Client的迴圈裡的if (confirm == 'Y')這個判斷式下手。

首先，我限用sprintf\_s()這個格式化輸出到陣列寄出連接成功的訊息，然後再進入迴圈開始接收訊息，在判斷接收的訊息，client端是否已經離開連線了，離開的話則直接關掉socket再return，再用sprintf\_s()輸出到陣列(要寄出的字串)，再寄到每個有連線的socket的端點。

而client端的話，是建立連線再收取從Server端寄來的資料印出內容，接著進入迴圈，如果輸入exit則跳出迴圈結束程式，如果沒有跳出則輸入要寄出的內容，我是用scanf\_s()(不能讀空白)，再寄出資料，之後再接收另一個Client的資料(先由Server接收再寄到別的Client)，而兩個Client接收跟寄出是相反的。

**程式碼**

**Server:**

DWORD WINAPI RunForClientThread(LPVOID input\_sIndex) {

char sendbuf[200], recvbuf[200];

int sockIndex = \*(int\*)input\_sIndex;

int i = 0, ret;

//傳送訊息給 client 端

//strcpy\_s(sendbuf, "");

//接收 client 端的訊息,

sprintf\_s(sendbuf,"Connection success!! you are Client(%d)",sockIndex);

send(connections[sockIndex], sendbuf, (int)strlen(sendbuf), 0);

while (true) {

ZeroMemory(recvbuf, 200);

ret = recv(connections[sockIndex], recvbuf, sizeof(recvbuf), 0);

if (strcmp(recvbuf, "exit") == 0 || ret == 0) {

printf("The client(%d) is disconnecting...", sockIndex);

closesocket(connections[sockIndex]);

return 0;

}

printf("The message from client(%d): %s\n", sockIndex, recvbuf);

sprintf\_s(sendbuf,"Client(%d) said: %s",sockIndex,recvbuf);

for (i = 0; i < 10; i++)

if (connections[i] != 0 && sockIndex != i)

send(connections[i], sendbuf, (int)strlen(sendbuf), 0);

}

}

**Client:**

if (confirm == 'Y')

{

connect(sConnect, (SOCKADDR\*)&addr, sizeof(addr));

ZeroMemory(recvMsg, 200);

recv(sConnect, recvMsg, sizeof(recvMsg), 0);

printf("%s\n", recvMsg);

while (strcmp(sendMsg,"exit") != 0) {

printf("Enter Msg: ");

scanf\_s("%s",sendMsg,200);

send(sConnect, sendMsg, (int)strlen(sendMsg), 0);

//接收 server 端的訊息

ZeroMemory(recvMsg, 200);

recv(sConnect, recvMsg, sizeof(recvMsg), 0);

printf("%s\n", recvMsg);

}

}

**Client1**

if (confirm == 'Y')

{

connect(sConnect, (SOCKADDR\*)&addr, sizeof(addr));

ZeroMemory(recvMsg, 200);

recv(sConnect, recvMsg, sizeof(recvMsg), 0);

printf("%s\n", recvMsg);

while (strcmp(sendMsg, "exit") != 0) {

//接收 server 端的訊息

ZeroMemory(recvMsg, 200);

recv(sConnect, recvMsg, sizeof(recvMsg), 0);

printf("%s\n", recvMsg);

printf("Enter Msg: ");

scanf\_s("%s", sendMsg, 200);

send(sConnect, sendMsg, (int)strlen(sendMsg), 0);

}

}

**執行結果 -- 測試資料及結果**

Server:

Server starting...

server: got connection from new client. (1/10)

server: got connection from new client. (2/10)

The message from client(0): 安安

The message from client(1): 你好

The client(0) is disconnecting...

Client:

connect to server?[Y] or [N] --> Y

Connection success!! you are Client(0)

Enter Msg: 安安

Client(1) said: 你好

Enter Msg: exit

bye bye...請按任意鍵繼續 . . .

Client1:

connect to server?[Y] or [N] --> Y

Connection success!! you are Client(1)

Client(0) said: 安安

Enter Msg: 你好

**討論 -- 遭遇困難及解決過程**

一開始總是不知道從何下手，但只要詳細知道PPT大致上再說什麼，應該都比較能夠知道那裏需要被改，因為原來只有一個Server和Client而已實在不知道要怎麼弄成多個Client互相通訊，但是本來就已經完成Client和Server的連接，我只要從一個Client傳送訊息到Server，再由Server傳送訊息給所有的Client(除了傳送訊息的Client)。

另一個問題是Client似乎需要非同步，因為如果是同步的話她會卡在那邊一直等待訊息傳送過來，所以我是把兩個Client的程式碼寫成相反的，一邊要先傳送訊息另一邊就要先接收訊息，這樣才不會同時都在等待訊息傳過來，所以只能一邊傳一邊收。

**心得 -- 自由發揮**

雖然功能沒有很完善但是我還是認識到很多東西，後面查了好多C的非同步怎麼完成，但就是查不到ㄟ，只有C#的非同步比較多，所以我就用了別的方式，來實現差不多的原理，也是改了好久喔，但也了解到很多伺服器端的運作。