程式設計期末 project-----socket programing

組員:資管一賴柏霖,林奇宥,徐紹軒,張凱庭

Q1:為何我們會想做這個:

Ans:　我們原本是想做校園地圖,可是後來要做的時候發現其他組已經有人寫過了,我們就決定換一個主題,後來我們就想作去年宿營 casino 的記分板,但是我們發現我們不會兩台電腦的傳輸,所以我們問小傑要怎麼連接兩台電腦,就走了上 socket 這條路.

Q2:我們在做的過程中遇到了甚麼困難:

ANS: 我們一開始就遇到了檔案無法執行的問題,因為網路上查到很多的 SOCKET CODE 編譯的環境都是在 LINUX 系統,所以我們找了很久才找到在 WINDOW 上編譯的 CODE,而且 DEV C++裡面的編譯環境也要多打一個參數才能編譯,後來我們建立了 1 台電腦對一台電腦的一對一連線,才想到我們應該要能多對一才對,於是我們又重新上網找資料,把原本的 CODE 刪掉,多用了 THREAD 來寫多對一的連線,至於怎麼做的,下面說.

SOCKET 簡介:

從網路的角度來看，socket 就是通訊連結的端點；從程式設計者的角度來看，socket 提供了一個良好互動的介面，使程式設計者不用知道下層網路協定運作的細節便可以撰寫網路通訊程式。

例如我們現在寫得在 windows 上的 socket 事實作於 Berkeley Software Distribution(BSD, release 4.3)中的 UNIX sockets 為基礎所發展出來的一套 API,而我們這次是用他其中的 TCP,但他也額外支援很多其他的格式,也提供電機碼的相容性, 因此在不同的作業系統上移植時並不需要做任何修改。

在 TCP/或 ip 的架構下,socket 可分為下面兩種傳輸方法

1. Datagram sockets:

   資料在 datagram sockets 間是利用 UDP 封包傳送，因此接收端 socket 可能會收到次序錯誤的資料,且部分資料可能會遺失,所我們這組採用下面的方法.

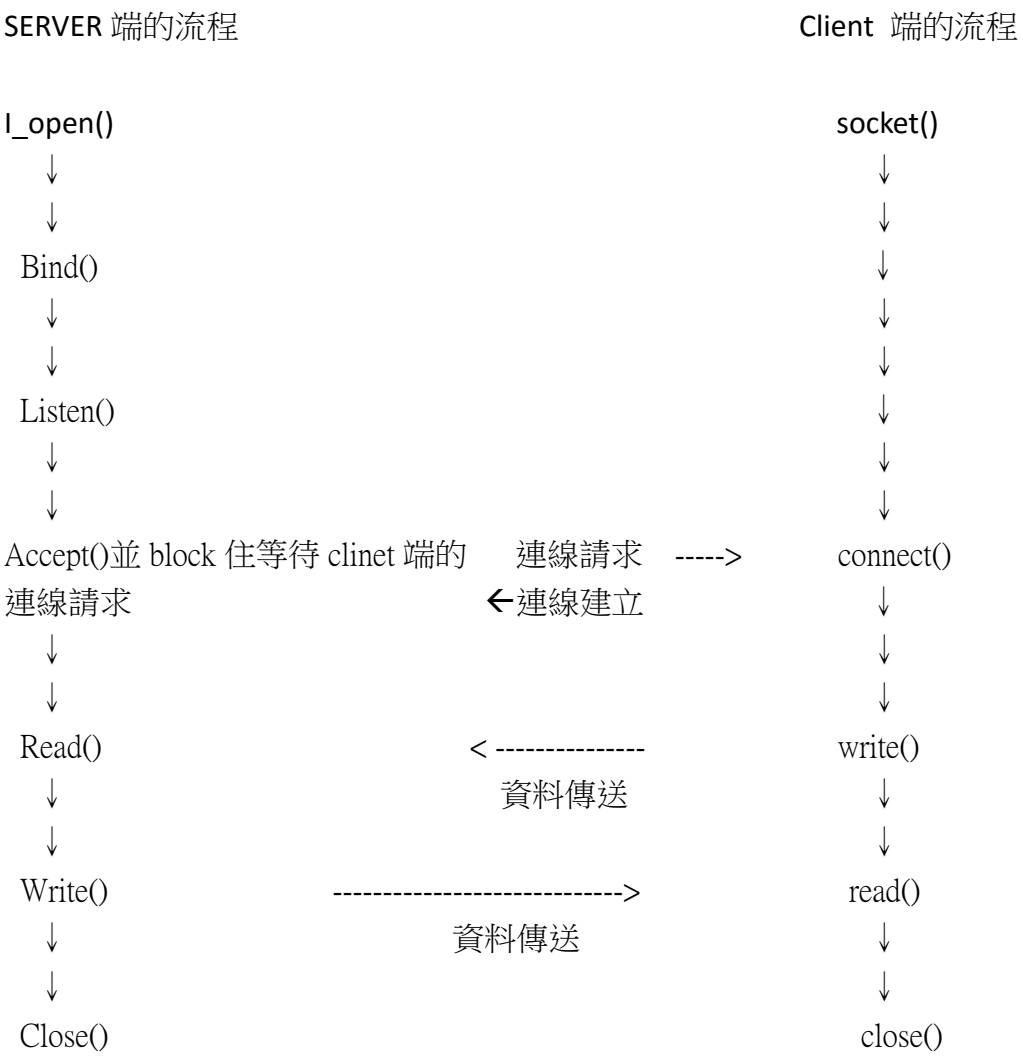
2. Stream sockets:

   資料在 stream socket 間是利用 TCP 的方式傳送, 因此接收端 socket 可以收到順序無誤、無重覆、正確的資料。此外 TCP 傳送時是採資料流的方式，因在傳送時會所有資料會視情況被分割在數個 TCP 封包中。

接下來是我們主程式架構的部分,上網查之後,發現所有網路應用程式皆可分為 5 個步驟:

1. 開啟一個 socket
2. 為 socket 命名
3. 與另外一個 socket 結合
4. 在 socket 間傳送資料
5. 關閉 socket

舉例來說



1. 我們開啟 socket 方法很簡單,只要呼叫 socket()函式就好.socket 是通訊網路的端點,就像介面卡,所以一個主機可以有很多 socket.而用戶端跟伺服器端都要有一個 socket 才能存取資料.

2. 接下來要為 socket 命名,為 socket 命名需要設定三個變數分別為(協定,連結埠號碼,位址),而用戶端就要用這些來與伺服器端連結,所以 server 端要先呼叫 bind()函式來設定好本身的連接埠號碼與 ip 位址.

3. 接下來要看如何將 socket 連結, 首先我們要先最少在 client 端跟 server 端都開了一個 socket,並且為 server 端的 socket 命名,接下來伺服器準備收封包,用戶端準備傳封包,這過程就叫結合,winsockapI 提供了幾個函式,server 端要開 listen(),收到連線要求後,就會開一個新的 socket,用 accept()跟他連結,而用戶端則是呼叫 connect()來連結.

4. 接下來我們在 socket 間傳送資料,如果兩個 socket 上,我們可以用 recv() 跟 send() 來傳送資料,然而我們有查到如果沒有連接兩個 socket 的話,也可呼叫 recvfrom()與 sendto(),雖然這個我們不會用也不知道為甚麼 QQ.

5. 接下來就是關閉 socket,其實這步很簡單,就是呼叫 closesocket(),關閉後可以將資源還給協定堆疊.

再來我們面臨的就是多人同時傳送資料給主機,因為我們現在的 code 都是執行緒一次往下讀,所以我們可以呼叫 thread 函式庫,來引用 thread.每當呼叫一個 thread,他就會新開一個背景執行緒,重頭往下跑,所以這樣就可以完成多人同時對一個 server 端傳送的工作.

後記: 我們原本只打算做賭場記分板的工作,不過後來因為還有時間,我們就又寫了 blackjack 這個遊戲,遊戲本身不算難,就是用 class player 這樣來分玩家,並把檔案存在 server 端,後來再把每個步驟按照順序丟在上面建立好的 socket 系統中,我們就寫好了多人連線的 blackjack.

心得:

因為我們本來的東西規劃好的想說蠻簡單的,所以打算到後來在做,沒想到後來別組先做出來了,所以我們在最後一周才開始趕工,不過因為卡到很多期末考,尤其是微積分,讓我們寫起來備感艱辛,這裡要特別感謝其他三位組員大力的 carry 我,雖然我門是四個人一起研究 socket,但是我到這周末才學會 socket 跟 thread 怎麼用 ,後來又面臨到把我們原先寫好的東西放到建立好的 socket 系統裡,但又出了不少差錯,因為 socket 系統我們還不是很熟悉,

所以有時候在傳封包的時候,會有一些奇怪的 bug 跑出來,像是我們在比完之後要攤牌把牌印出來之後,就出現了有些人印不出來的 bug,總之,這次的 project 我們真的學習了許多,上網查了沒用過的函式庫,大家在一次的體驗一起寫 code,真是好的經驗.

.