**Final Project of Computer Networking**

1. Implementation
2. Server-side

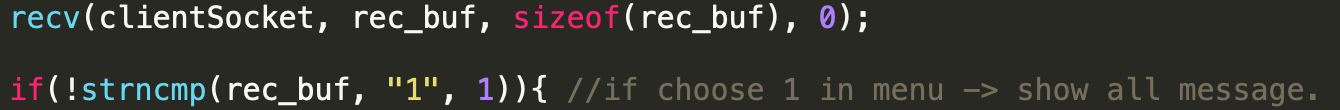
Server-side要做的事情是與Client連接，然後送出資訊告訴Client可以執行哪些動作，最後接收Client傳過來的資訊並做處理。

在連接方面，我的實作與範例code有所不同，我將accept function放於While-loop之前，並把While-loop內的closesocket function移至loop之後，讓Client與Server不需要在每次傳遞封包後就重新連接。

在判斷、資料傳送與資料儲存方面，我總共使用了4個char array，分別是menu[MAX\_SIZE], all\_message[MAX\_SIZE], send\_buf[MAX\_SIZE], rec\_buf[MAX\_SIZE]。

一張含有 文字 的圖片

自動產生的描述send\_buf是用來記錄要傳送至Client的訊息，並使用send function將資料send\_buf內容傳至Client-side（使用方式如下圖）。

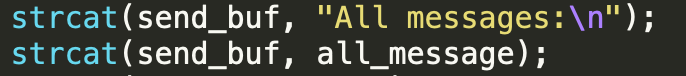
rec\_buf則是用來接收Client傳過來的訊息。並可以使用strncmp去判斷要執行哪一個動作（使用方式如下圖）。

一張含有 文字 的圖片

自動產生的描述menu的內容是固定的，它會存著menu訊息，再藉由send\_buf將menu資訊傳至Client（menu的內容與使用如下圖）。

一張含有 文字 的圖片

自動產生的描述最後一個all\_message是用來記錄曾經輸入過的message。在menu選擇option 2之後，可以在Client-side輸入訊息，Server-side會將接收的訊息放入all\_message（如下圖）。

一張含有 文字 的圖片

自動產生的描述如果選擇option 1，會將all\_message的內容接於send\_buf，並於之後傳至Client-side（如下圖）。

1. Client-side

Client-side這邊是負責主要的操控。在連接Server後，會將Server傳過來的資訊顯示於螢幕上，方便使用者知道當下應該要做什麼操作，並在完成操作後將資料傳至Server端。

與Server-side相同，我將連接用的accept function放在While-loop之前讓兩端先做連接。

一張含有 文字 的圖片

自動產生的描述在While-loop裡，Client-side要做的事情只有印出Server傳過來的訊息與發送訊息給Server（如下圖）。

1. Screenshot and explanations

一張含有 文字 的圖片

自動產生的描述下圖為Server-side啟動後第一步，還沒有與Client連接，維持在waiting。此時只要Client端發送連接要求，兩邊就可以開始連線。

一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述接著一張含有 文字 的圖片

自動產生的描述client-side就送出連接的請求。在連接上之後Client的介面會出現從Server送過來的menu，而Server介面則顯示Connect Successfully。

接著測試各項功能是否正常，如果在menu輸入’1’就會依序練出目前存在的message，並以a,b,c…排序。如果在menu輸入’2’會請Client輸入new message。如果輸入’exit’雙方就會斷開連接（下列依序列出實際測試狀況）。

一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述

1. Difficulties and solutions

這份作業遇到的最大問題是電腦環境的適應。因為平常使用的是mac，在使用windows編譯很不習慣，而且在windows的command line中，沒有辦法使用gcc compile檔案。為了解決問題，上網找了許多插件像是mingw之類，在耗費多時後，終於成功在windows上操作。雖然環境不熟悉，但我認為身為一個資工系學生，快速適應不同電腦環境以及command line的使用是必備的技能，這次能有這樣的收穫十分可貴。

另外，這次用上了winsocket2這個函示庫。雖然大部分的code老師都已經提供，但畢竟以前沒有使用過，在一開始要看懂範例code時也花費了些許時間，但因為網路上資料庫內容充足且詳細，在之後的使用及改寫非常的順利。