**電腦網路實驗實驗報告 < Socket Programming >**

**姓名: 翁佳煌　　　學號: 409430030**

# 實驗名稱

網路程式設計，寫一個會標記「已讀」訊息的echo server，並寫一支client程式跟echo server對話。並將上次實驗的IPv4 TCP echo server和client改成IPv6。

# 實驗目的

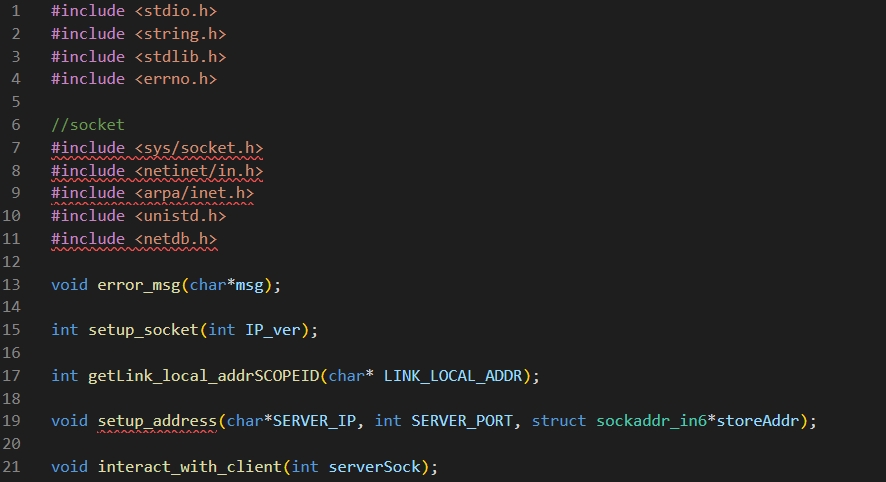
這次實驗將讓我們理解通訊協議、網路設備、網路架構中的client/server模型以及 TCP socket programming的基本概念，讓我們對網路程式設計有整體的認識，並通過 getaddrinfo 和 TCP client/server 的示範演示來演示如何實現這些技術。通過本實驗的學習，將學會如何建立基於 TCP 協議的客戶端/伺服器應用程式，並掌握如何使用網路編程的基本工具和API。

# 實驗設備

Linux作業系統之電腦。

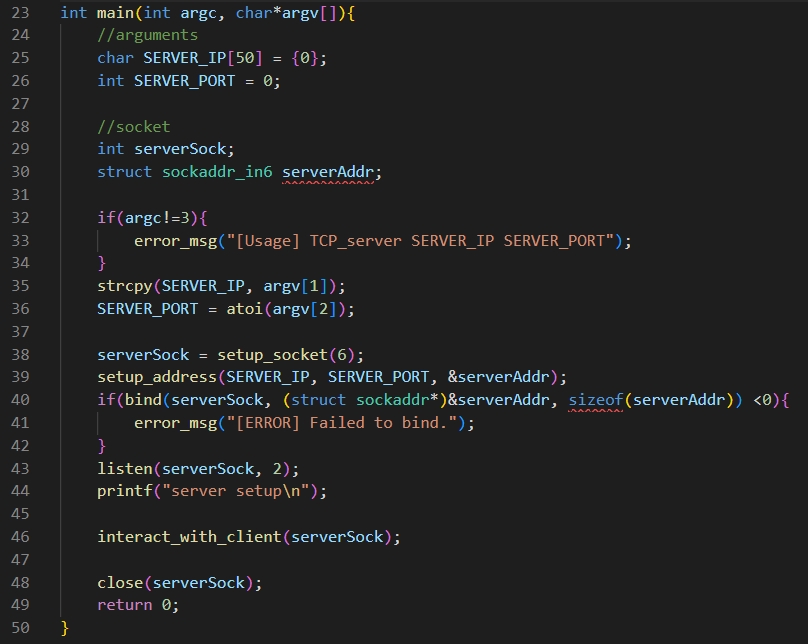
# 實驗步驟

首先看到**server端**，它會持續等待客戶端的連線，並接收、處理客戶端的訊息並回覆已讀給client，以下是詳細code介紹:



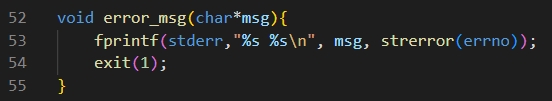
1~11行: 引用需要用到的標頭檔，除了c語言常見的標頭以外，還包含sys/socket.h、netinet/in.h、arpa/inet.h、unistd.h等等。

12~21行: 宣告函式，分別為error\_msg、setup\_socket、setup\_address、getLink\_local\_addrSCOPEID、interact\_with\_client。



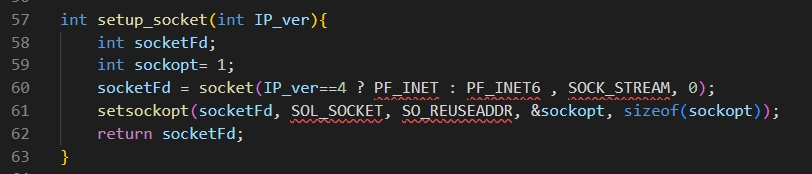
23~50行:

這段程式碼是主程式main，首先宣告了變數SERVER\_IP和SERVER\_PORT，並初始化為0。接下來宣告了一個整數型別的變數serverSock和一個結構體型別的變數serverAddr。透過argc和argv判斷傳入的參數是否符合程式所需的兩個參數，如果不符合則輸出使用說明。之後，將argv[1]指向的字串複製到SERVER\_IP陣列中，並將argv[2]轉換成整數型別的SERVER\_PORT。接著，呼叫setup\_socket函數，傳入6作為IP版本的參數，取得一個socket描述符並指派給serverSock變數。之後呼叫setup\_address函數，傳入SERVER\_IP、SERVER\_PORT和&serverAddr三個參數，該函數設置了serverAddr的成員變數。最後，呼叫bind函數，綁定socket描述符和serverAddr結構體，如果綁定失敗，輸出錯誤信息並退出程式。接著呼叫listen函數，告訴操作系統這是一個服務器端的socket並設置可以同時連接的客戶端的最大數量為2。最後輸出"server setup"，表示服務器準備就緒，準備開始接收客戶端的連接請求。最後呼叫interact\_with\_client函數，傳入serverSock描述符，開始和客戶端進行交互。交互結束後，關閉socket描述符並退出程式。

52~55行:

用來輸出錯誤訊息到標準錯誤輸出 stderr。該函式接受一個指向字串的指標 msg 作為參數，代表需要輸出的錯誤訊息。

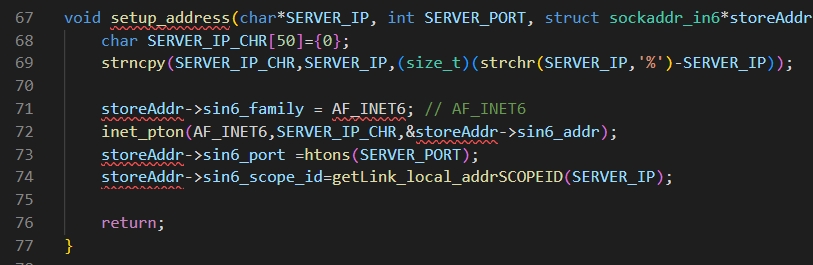
在函式內部，使用 fprintf 函式將 msg 和使用 strerror 函式獲取的錯誤訊息（由 errno 變數表示）格式化輸出到標準錯誤輸出中，然後使用 exit 函式退出程式並返回錯誤碼 1，以便告知呼叫該函式的程式出現了錯誤。



57~63行:

這個函數是用來創建一個socket的，接受一個整數參數 IP\_ver，用於指定要創建的socket的 IP 地址版本。如果 IP\_ver 為 4，則使用 PF\_INET 創建一個 IPv4 socket；如果 IP\_ver 為 6，則使用 PF\_INET6 創建一個 IPv6 。

接下來，創建一個名為 sockopt 的整數變量，並將其設置為 1。然後使用 setsockopt() 函數將 SO\_REUSEADDR 選項設置為socket。這將使操作系統在socket關閉後立即釋放socket的本地地址，以便在同一地址上重新綁定socket時不會發生地址已被使用的錯誤。最後，返回socketFd。



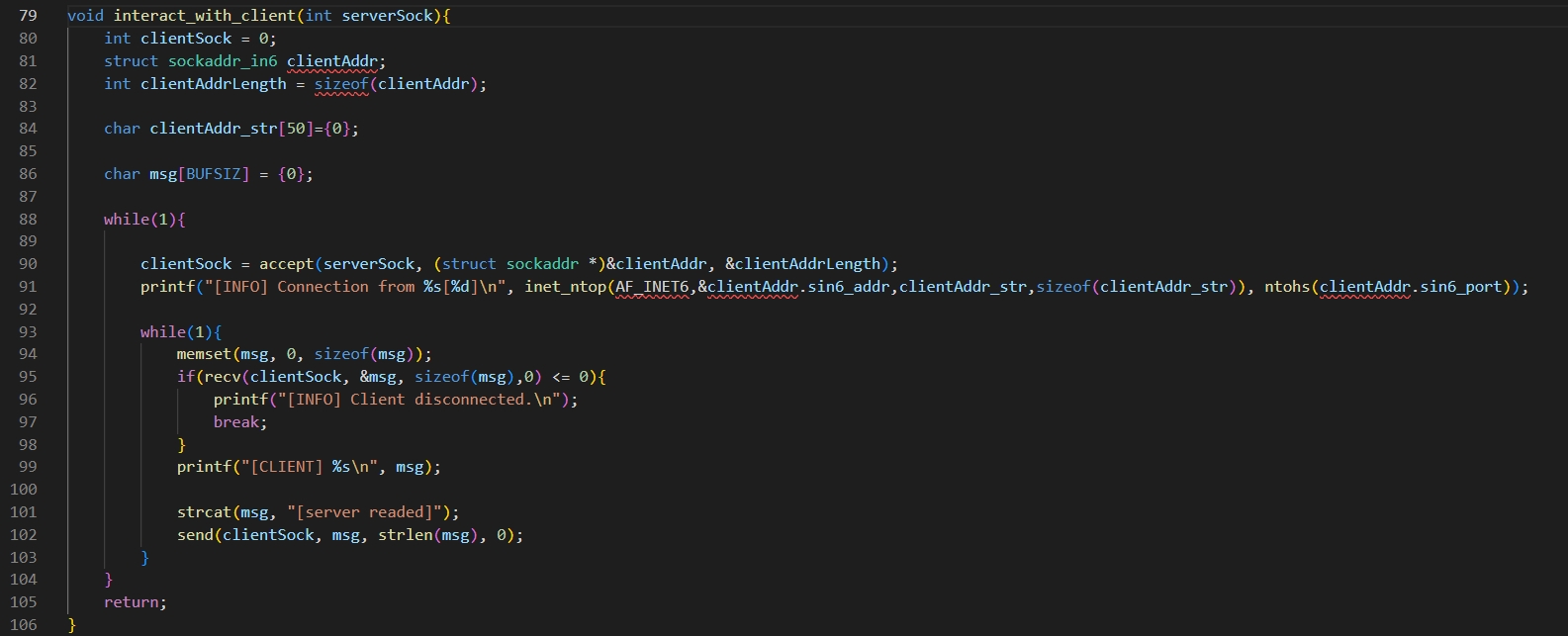
67~77行:

這段程式碼的功能是為了設定伺服器端的位址資訊，其中需要傳入 SERVER\_IP (伺服器 IP 位址) 與 SERVER\_PORT (伺服器端的連接埠號碼)，以及一個存儲設定好的位址的結構指標 storeAddr。

首先，程式碼會先宣告一個大小為 50 的字元陣列 SERVER\_IP\_CHR 並初始化為 0。接著，使用 strncpy() 函式將 SERVER\_IP 中的字元複製到 SERVER\_IP\_CHR 中，直到遇到 '%' 符號為止，這是為了去除 IP 位址中可能存在的範圍識別碼 (scope identifier)。

然後，程式碼會設定存儲位址的結構指標 storeAddr 中的成員。storeAddr->sin6\_family 被設定為 AF\_INET6 表示使用 IPv6 協議。inet\_pton() 函式可以將字串格式的 IPv6 位址轉換成網絡字節序的二進制形式，將其儲存到 storeAddr->sin6\_addr 中。storeAddr->sin6\_port 被設定為 SERVER\_PORT 的值，並且使用 htons() 函式進行大小端轉換，將其轉換為網絡字節序。

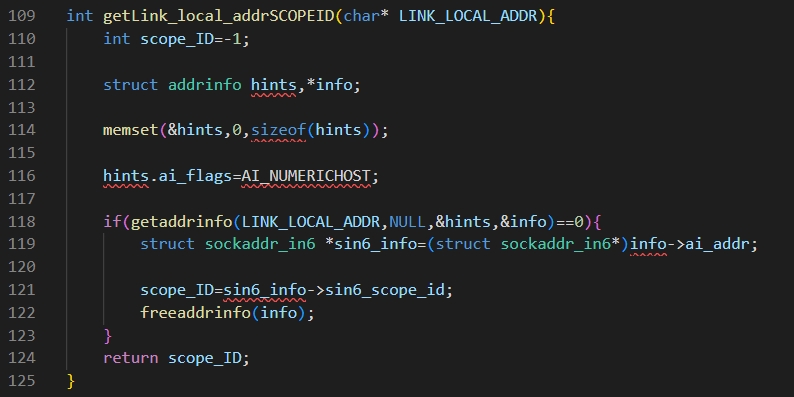
最後，使用 getLink\_local\_addrSCOPEID() 函式來取得該位址的範圍識別碼，將其存儲到 storeAddr->sin6\_scope\_id 中。最終設定好的位址結構指標 storeAddr 就可以被用來綁定到 socket 上了。

79~106行:

這個函式 interact\_with\_client 是用來處理與客戶端之間的互動。在函式開始時，會接受一個參數 serverSock，這個參數是代表伺服器端的 socket，也就是在主函式中所建立的那個。接著，宣告一個 clientSock 變數來儲存與客戶端連線的 socket，並且也宣告一個 clientAddr 變數，用來儲存客戶端的地址。透過 accept 函式可以接受客戶端的連線，同時取得客戶端的地址，並且建立一個新的 socket 來處理和這個客戶端的連線。

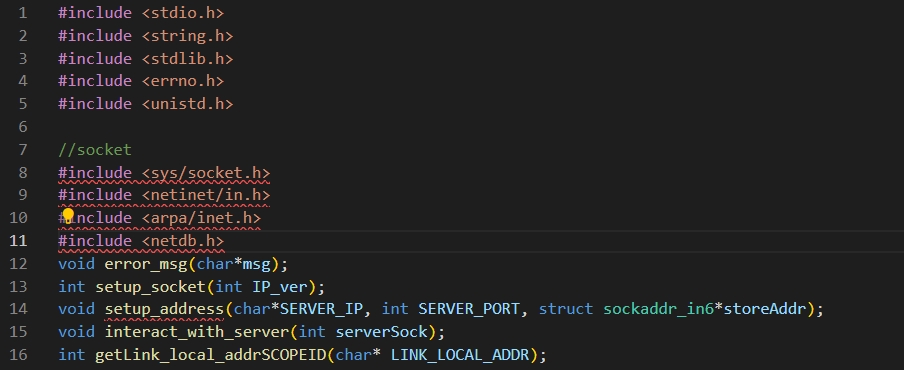
接著進入無窮迴圈，直到接受到客戶端的斷線訊息為止。在這個迴圈中，會先宣告一個 msg 變數來儲存從客戶端傳來的訊息，然後使用 recv 函式來接收客戶端的訊息。如果接收到的長度小於等於 0，表示客戶端已經斷線了，此時會顯示一個訊息，並且跳出迴圈。否則就會顯示客戶端傳來的訊息，並且在訊息後面加上一個字串 "[server readed]"，然後透過 send 函式把這個訊息送回給客戶端。

總結來說，這個函式主要是用來處理和客戶端的連線，接收客戶端的訊息，並且回應一些訊息給客戶端。

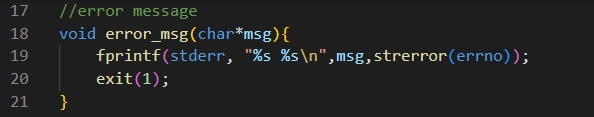
109~125行:  
這個函式是用來從給定的 Link-local IPv6 位址中提取其對應的 Scope ID。在 IPv6 網路中，Link-local IPv6 位址是一種在單個鏈路範圍內使用的位址，它的格式是 fe80::/10。Scope ID 則是一個用於區分不同網路界面（例如網卡）的整數值。

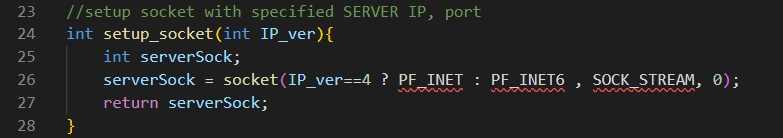
函式接收一個 Link-local IPv6 位址的字串作為參數，然後使用 getaddrinfo() 函式將其轉換為一個 addrinfo 結構，然後從中提取 Scope ID。該函式使用了 struct sockaddr\_in6 類型，該類型表示 IPv6 地址和端口號。在這個函式中，為了讓 getaddrinfo() 函式可以識別 Link-local IPv6 位址，使用了 ai\_flags 參數設置 AI\_NUMERICHOST 標誌。最後，該函式返回提取的 Scope ID。

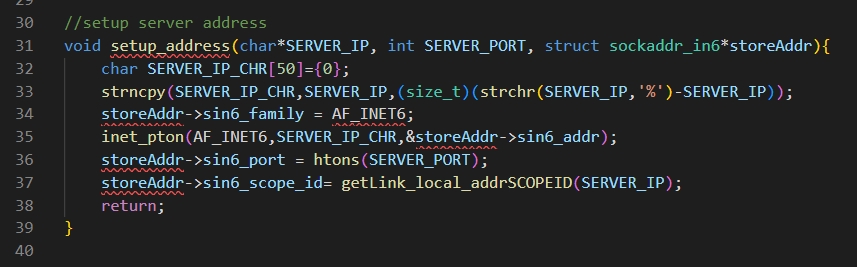
接下來是介紹**client端**，它需要輸入訊息並傳送到server端後，要接收到server端送來的已讀訊息，以下是詳細code介紹:



1~16行: 引用需要用到的標頭檔，除了c語言常見的標頭以外，還包含sys/socket.h、netinet/in.h、arpa/inet.h、unistd.h等等，並宣告函式，分別為error\_msg、setup\_socket、setup\_addressget、Link\_local\_addrSCOPEID、interact\_with\_server。



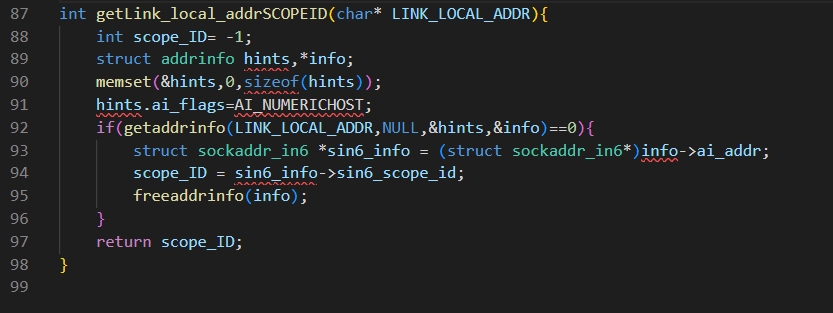




17~40行:這幾行都與server端的部分一模一樣，這裡就不再重複敘述一次。

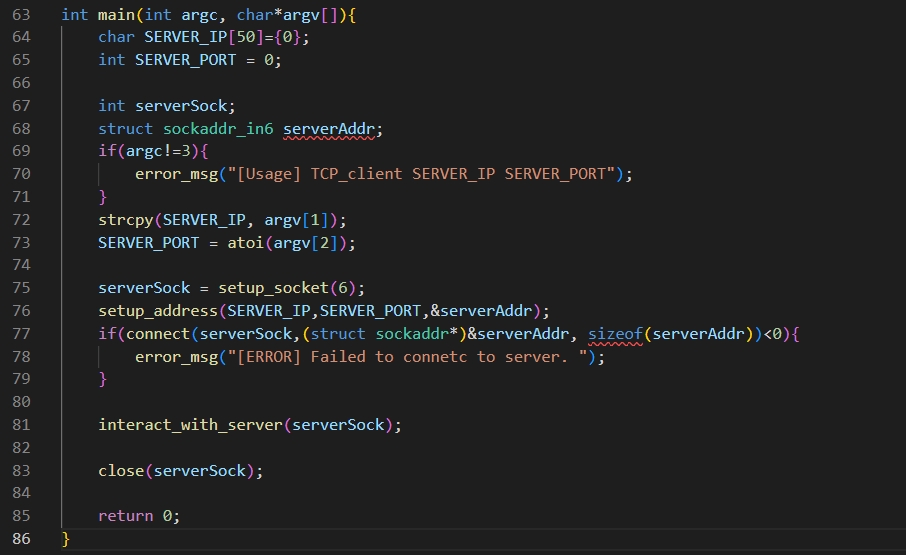


41~61行:也與server端的interact\_with\_client很類似，這個函數 interact\_with\_server 是用來和服務器進行交互的。函數會一直進入一個無限迴圈，並且在每次迴圈中等待用戶輸入信息，然後將信息發送給服務器。如果成功發送信息，則函數會等待接收服務器回傳的信息，並在控制台上印出服務器回傳的信息。如果接收失敗，函數就會跳出迴圈並結束函數。



87~99行:

這個函數主要是用來從傳入的 LINK\_LOCAL\_ADDR 參數中獲取該 IPv6 位址的 SCOPE\_ID，即該位址對應的網路接口編號。在函數中，首先使用 getaddrinfo 函數來獲取 LINK\_LOCAL\_ADDR 的地址資訊，接著從該地址資訊中取出 sockaddr\_in6 結構體指針 sin6\_info，透過 sin6\_scope\_id 成員讀取 SCOPE\_ID。最後使用 freeaddrinfo 釋放獲取的地址資訊。如果獲取 SCOPE\_ID 成功，該值會被返回，否則返回預設值 -1。

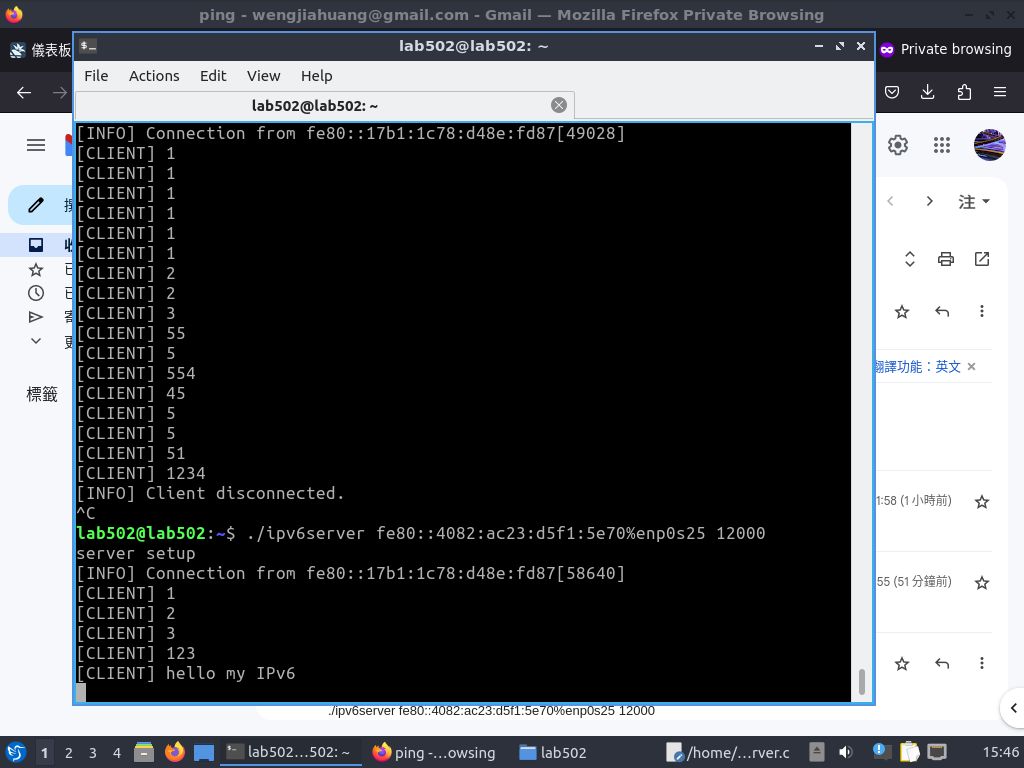


63~86行:  
這裡是client的main函式，會從命令列參數中讀取使用者所提供的 SERVER\_IP 和 SERVER\_PORT。然後使用 setup\_socket() 函式建立一個 IPv6 的 socket，使用 setup\_address() 函式初始化 sockaddr\_in6 結構變數，包含 SERVER\_IP 和 SERVER\_PORT，並設定 scope\_id。最後，使用 connect() 函式連接到 server，如果連接失敗則呼叫 error\_msg() 函式顯示錯誤訊息。

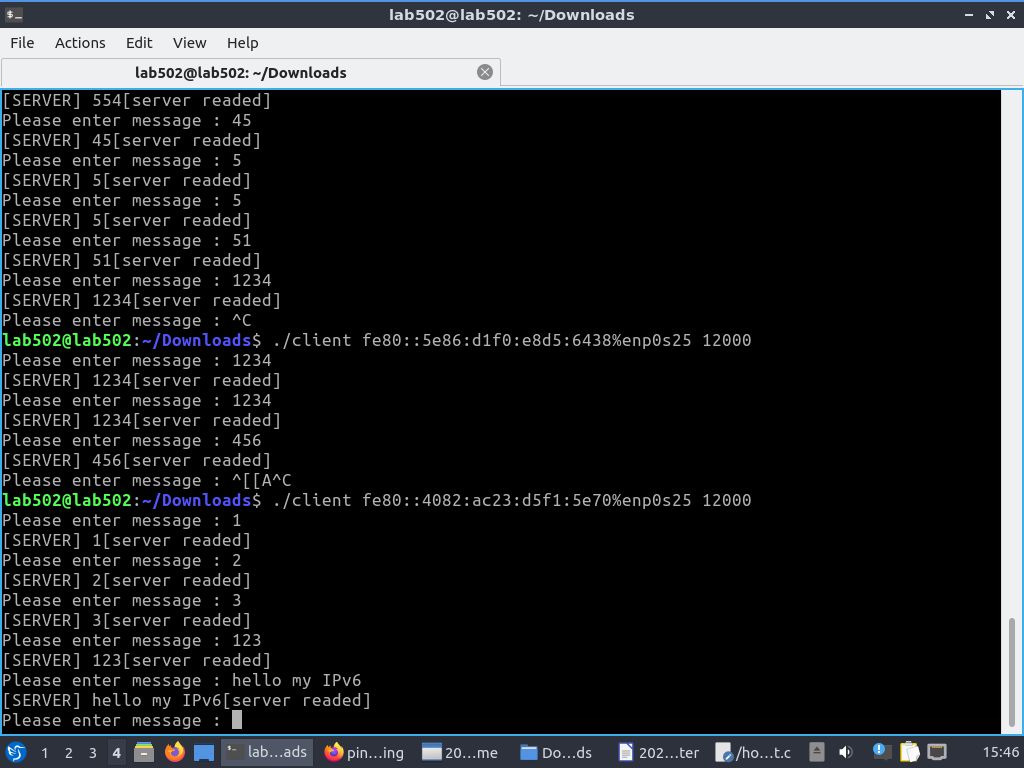
接著，呼叫 interact\_with\_server() 函式，與 server 進行互動，將使用者輸入的訊息傳送到 server，並顯示 server 回傳的訊息。當與 server 的連線中斷時，透過 close() 函式關閉 socket。最後回傳 0 表示程式執行成功結束。

以下是測試結果:

Server:



Client:



# 問題與討論

1. 有許多ipv6的函數仍然需要再更進一步理解，包括其用途以及需要的參數。
2. 如果改使用ipv6有許多地方需要更動，例如將 PF\_INET 改成 PF\_INET6，以支援 IPv6等等。這問題在下禮拜的實驗課將會實作。
3. Socket Programming 在 Linux 和 Windows 上的差異，包括Socket 創建方式不同、IP 位址與埠號表示方式不同、Socket 的清除方式不同等等。

4. IPv6的部署比IPv4慢，並且一些網路還不支持IPv6。這意味著，在某些情況下，使用IPv4可能更為可靠或更具成本效益。

# 心得與感想

這次的實驗讓我更深入地了解了Socket Programming的概念，這次使用了IPv6的Socket Programming，相比IPv4，IPv6具有更大的地址空間、更好的安全性和更高的效率等優點。但是，IPv6在實際使用中仍然存在一些問題。

首先，IPv6的普及率相對較低，因為目前仍有很多應用程序和硬件設備只支持IPv4。這可能會使得IPv6應用的普及受到限制。其次，IPv6的地址格式相對複雜，需要進行較為複雜的設置和配置，對於一些不熟悉IPv6的用戶來說，可能會造成一定的困擾。此外，在實際應用中，IPv6網路環境的構建和配置也需要進行相應的調整和優化，以提高網絡的可靠性和效率。

總體而言，IPv6作為一種新型網際網路協議，具有諸多優勢，但在實際應用中仍面臨著一些挑戰和問題。通過本次實驗，我們能更深入了解IPv6的基礎知識和原理，也能更好地理解IPv6的應用前景和發展趨勢。

# 參考文獻

<https://zh.wikipedia.org/zh-tw/IPv6>

<https://test-ipv6.com/index.html.zh_TW>

<https://www.ithome.com.tw/tech/92046>

<https://ipv6.twnic.tw/about-inro.html>

<https://hackmd.io/@ki_2CYV1QfCsYx3X8THqiw/pty11111>

<https://nordvpn.com/zh-tw/blog/ipv4-ipv6-qubie/>