計算機科學概論第二次作業

網路程式分析

內容

一、 何謂 Socket ？Socket 概念介紹

二、 小型伺服器程式 nweb.c 分析

三、 改良：關於我的 MiniWebServer

四、 結論與心得

頁數較多的報告，加一個封面和目錄頁是好習慣，不過這份報告的段落名稱取的太過口語化了。(一)和(三)可以修改如下：

一、Socket 概念介紹。

三、改良的 MiniWebServer 討論。

另外，最前面應該有一個小段落做報告目的簡介。

國科會研究報告規範的標準字型，
中文要用"標楷體"，英文是 Times New Roman.
一般報告中文也可以用細明體，其它的就儘量不要用.

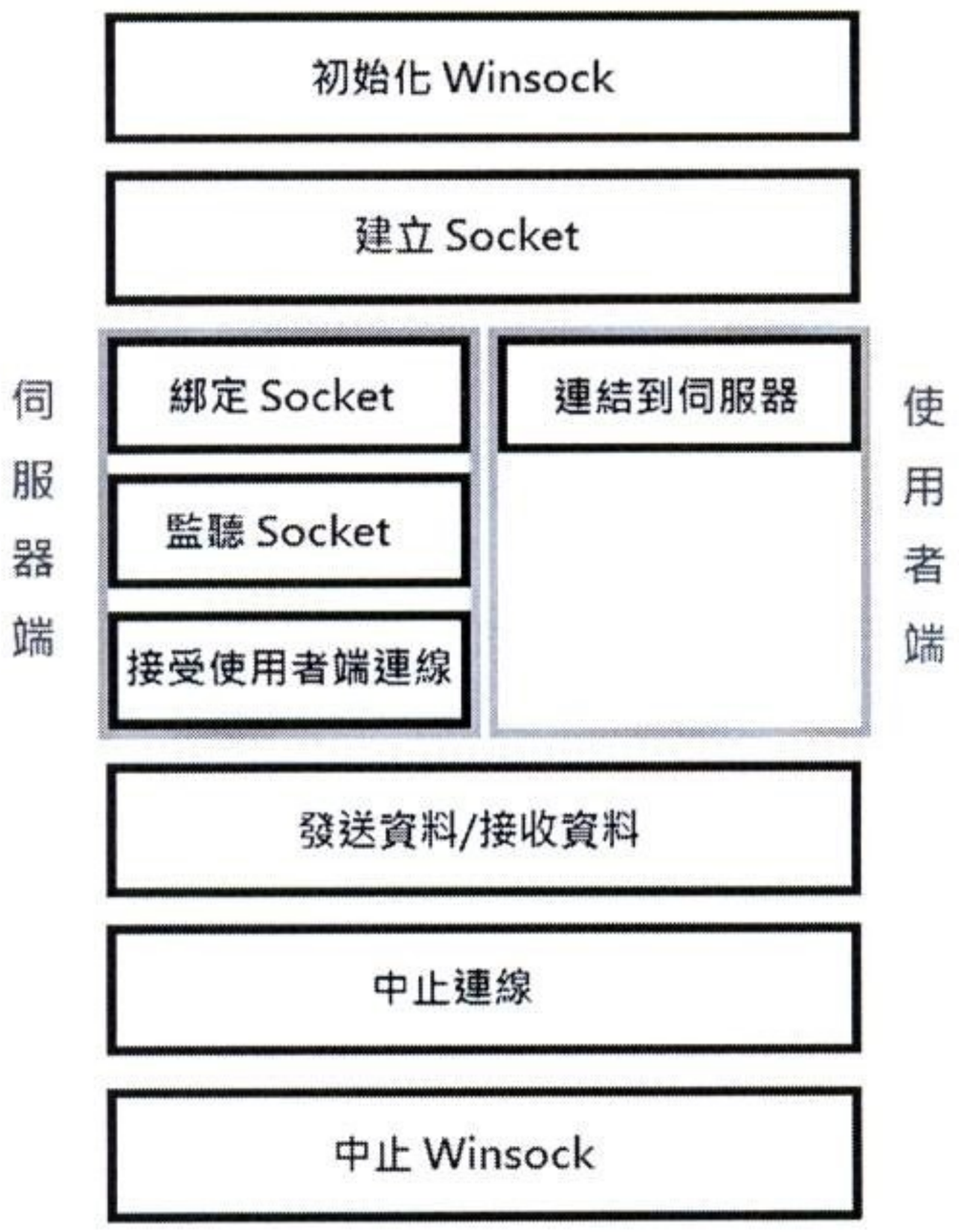
一、何謂 Socket ？ Socket 概念介紹

Socket 是在 UNIX 環境下一個網路通訊的介面，最早源於 BSD 上的 Socket 程式庫，後來在做了一些小修改之後被加入到 POSIX 的標準中，才變成 UNIX 上通用的 API 。

在 Socket 程式設計裡用的是 Server-Client 模型，兩台通訊的電腦中會有一臺扮演伺服器端，另一臺則扮演使用者端（大家好像叫「客戶端」的比較多）。兩方所做的事情大致如下圖所示（以 Winsock 為範例）：

報告的排版太鬆散，除非這是報告的初稿，如果是定稿，不應該用 double-space.

太口語化，應該寫：（或稱「客戶端」）



一開始，程式必須要和需要用到的動態程式庫連結，以便於使用

報告（特別是頁數多的）一定要加頁碼.

位於其中的函數。之後，就要建立 Socket 來處理連線，使用者端會建立一個 Socket ，伺服器端會建立二個 Socket 。伺服器端和使用者端都會有一個 Socket 用來發送、接收資料，伺服器端則多一個 Socket 用來監聽連接埠。

第二部分，伺服器端必須使用一個 Socket 來綁定一個連接埠，並且決定使用的通訊協定。接下來就要綁定一個連接埠並且監聽有無連入連線的請求。若有連入的請求，則建立一個 Socket 和使用者端連線。使用者端則是發送連線請求到伺服器端，若伺服器端接受請求，那這時便已建立好通訊用的 Socket ，可以開始發送、接收資料了。在這步驟中使用者端和伺服器端都必須解析位址的資訊，不過伺服器端是解析要綁定的連接埠或服務（例如 http），使用者端要解析伺服器端的域名（要查詢 DNS）和連接埠或服務。

在發送、接收資料完成之後，便需要關閉連線（shutdown）、關閉 Socket。最後，若不須用到 Socket 時，則把連結的動態程式庫釋放掉，完成這一整個流程。

二、 小型伺服器程式 nweb.c 分析

前置處理 / 巨集

開頭的部分，定義了一些常數、要引入的標頭，值得注意的是這一句 `#pragma comment(lib, "WS2_32.lib")` ，這句是告訴 Visual C++ 要將程式與 WS2_32.lib 此靜態程式庫連結， Socket 的函數都被定義在裡面。若使用 MinGW 則必須在連結階段加上參數 `-lws2_32` ，將程式與 libws2_32.a 連結。

全域函數、結構

程式一開始在全域定義了一個匿名的結構，裡面有兩個字元指標陣列成員，一個叫做 ext ，用來儲存延伸檔名，另一個叫做 filetype 則是用來儲存各 ext 所對應到的 MIME Type ，在傳送檔案時必須告知使用者端一個檔案的 MIME Type。

另外全域亦有三個函數 syslog 、 web 和 main 。 syslog 負責將紀錄存到 nweb.syslog 檔案中， web 則負責處理使用者端的請求。 main 則在前置作業完成之後持續等待連入的請求，接受到一個請求之後再呼叫 web 來處理 http 的請求。

程式流程

程式一開始，先切換工作目錄到 C:\public_html ，並且呼叫 WSAStartup 以和需要用到的動態程式庫連結。之後解析本機位址，選擇使用 IPv4 和 TCP ，並和本機的 80 埠綁定。之後開始監聽，並允許最大 64 個連入請求。

接下來是主要的部份，在 main 接受一個連入請求之後，呼叫 web 函數， web 函數會接收 http 請求並且解析；若 http 的請求為 GET 且檔案存在，則告訴對方該檔案的 MIME Type 並且傳送。若用了非 GET 模式、請求父目錄的檔案、不支援的檔案類型或是檔案不存在則會在寫入紀錄之後結束程式。

這過程中有一個小細節是，若是請求為網頁的小圖示（favicon.ico）則告訴對方這邊沒有該圖示。

程式流程最好是用流程圖(或UML)分析.

三、 改良：關於我的 MiniWebServer

nweb 的缺點：原始碼層面

首先必須指出幾個在原始碼層面的問題。第一個是在 280 行的此句：

```
socketfd = accept(listenfd, (struct sockaddr *) &cli_addr, &length);
```

length 的型態是 size_t 但是傳入的必須要是 int 指標，在 64 位元編譯器下 size_t 的大小有可能是 8 位元組！第二個是在 253 行的：

這句話有問題，在64位元的編譯器下，int 也是8位元組。

```
if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
```

socket 傳回值為 SOCKET 型態，為無號的整數，所以這個比較是無意義的。比較好的做法是把 listenfd 轉換成 int 之後再做比較。

這句話是錯的，socket() 傳回的是 I/O descriptor 是 int 的型態。

第三個老師上課曾經說過的在第 100 行的：

```
nbytes = recv(cli_socket, buffer, BUFSIZE, 0);
```

若 recv 為非阻塞模式呢？在 Windows 下預設為阻塞模式，但是我認為在有些系統下為非阻塞模式，無論有無收到資料都會直接傳回。

socket 預設都是阻塞模式，要透過 ioctl 才能改變為非阻塞模式。
這行敘述的問題在於 tcp 不會保證每一筆 client 傳來的資料都能用單一一個 recv() 收完整。

nweb 的缺點：功能層面

我不太喜歡 newb 一遇到錯誤就立刻結束，而且所記錄的資訊不夠詳實。紀錄中可以加上時間，並且在接受連入請求時也應該要記錄對方的 IP 。在呼叫 accept 時第二個參數就包含了對方的資訊，可以從裡面挖出 IP 的資訊。再者， nweb 所支援的格式太少。另外， nweb 沒有謹慎地檢查錯誤，例如發送、接收時可能會出錯。

改良：關於我的 MiniWebServer

我的 MiniWebServer 是依照 nweb 來改寫的，在作業附檔中可以找到原始碼，以我以前寫的 SocketIO 程式庫來處理瑣碎的問題來讓原始碼的可讀性更好。 MiniWebServer 主要對 nweb 做了以下修改：

1. 紀錄更詳實：記錄連入的使用者端 IP，紀錄也有附上時間
2. 若遇到錯誤不會結束程式
3. 若非 GET 模式會告知對方 501 Not Implemented
4. 允許 favicon.ico ：若是 favicon.ico 真的存在呢？
5. 加入更多格式外，若不支援則用 application/octet-stream 取代
6. 傳送檔案時會告知對方檔案的大小

更多細節請看原始碼。(建議使用 MinGW 編譯)

五、 結論與心得

這個作業的程式看懂較非問題，問題在於要如何改良？在測試的過程中覺得 nweb 因為錯誤發生就直接結束讓它的方便度下降很多。在完成了改良之後，我把它當成方便的傳檔案工具，當要傳送檔案給對方時就可以很方便地把自己的電腦當成一個小伺服器！不需要再把檔案傳送到網路硬碟之類的空間外，可以直接用瀏覽器傳送也是最方便的一點。

另外在改良的過程中修正了我一些舊的錯誤觀念。例如在加入顯示對方 IP 之後發現使用者端的連接埠怎麼都不是 80 ？才發現連接埠並非一對一的。另外還有看到一些 http 的請求長什麼樣子，回應長什麼樣子。

→ 當然，要不然 NAT 就沒辦法運作了。

印象最深刻的是，有些瀏覽器（例如 Google Chrome）會在原本的連線完成之後多發送一個連入請求，占用一個連入的配額，直到一段時間之後才釋放，若是原本的 nweb，不知道這樣會不會造成它錯誤呢？

→ 這個實驗不難作，報告最好能寫出實驗結果。

總之，在做這個作業時，改良的過程讓我獲得了許多樂趣！