Project1

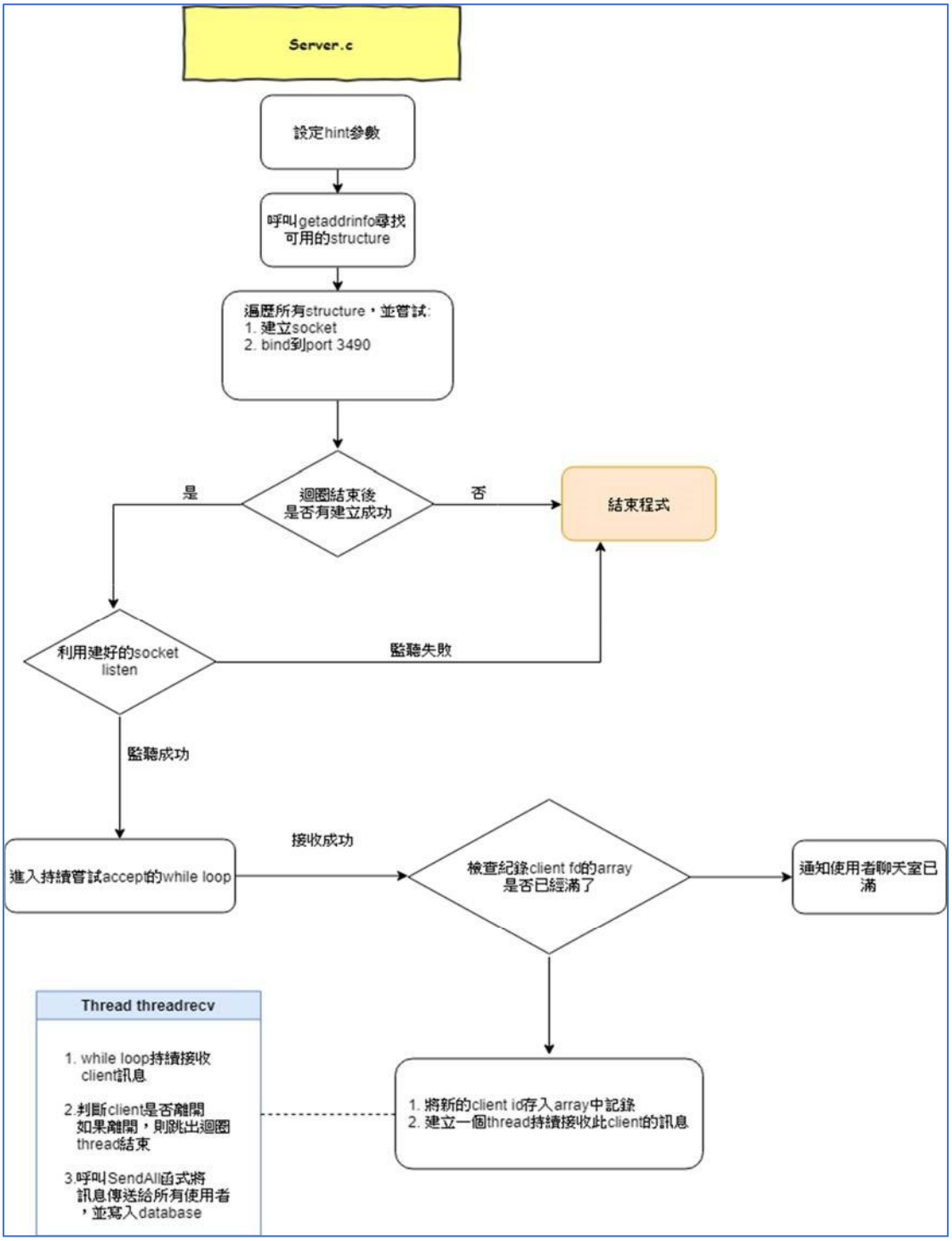
周易廷

⬤Describe the workflow and architecture of the system.

(一)多人聊天室

Server.c:

流程圖:



說明:

首先利用建立 addrinfo 結構 hint，並設定相關資訊，在此程式中設定了: 1. IP

2. IP family
3. TCP stream

接著利用函式 getaddrinfo 尋找符合條件的 structure，並利用 for 迴圈遍歷所有 getaddrinfo 回傳的 linklist，嘗試建立 socket 並 bind 到 port 3490，如果在遍歷完整個 linklist 後，沒有找到可以使用的 structure，則結束程式。

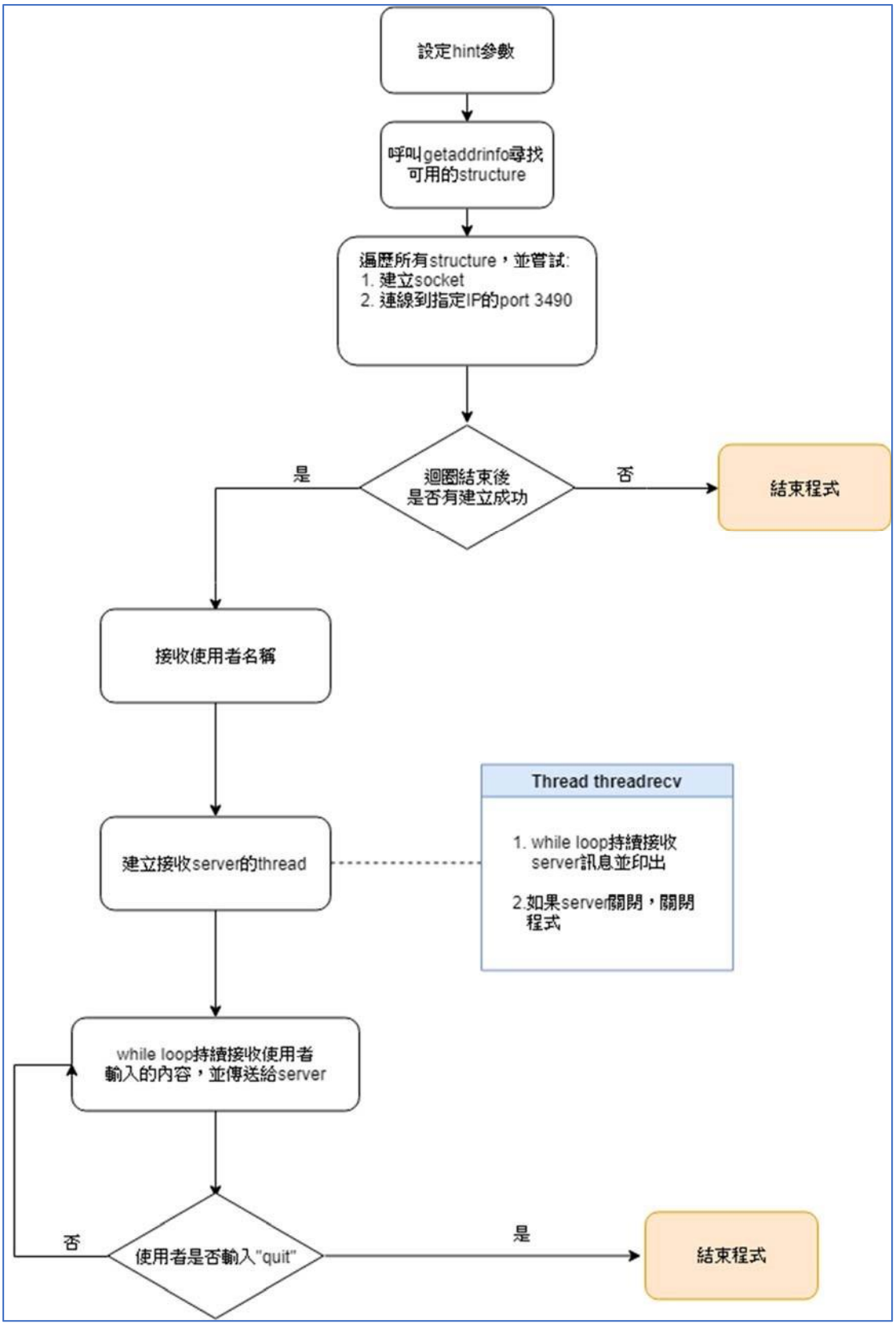
如果有成功建立出 socket，則用建立好的 socket 監聽，並利用 while loop 持續檢查是否有 accept 到 client 的 connection。

在接收成功的情況下，accept 函式會回傳一個檔案描述符，此時我們利用 array usesocks 將檔案描述符儲存起來。如果在儲存之前，array 已經滿了，就代表聊天室已經滿了，程式會 send 一個訊息給目前 accept 到的 client，假設聊天室還沒滿，此時程式將產生一個額外的 thread threadrec，專門用來接收來自這個 client 的訊息，在訊息傳來時，呼叫 SendAll 遍歷紀錄檔案描述符的 array usersocks，傳送訊息給所有 client。

Threadrec 功能:

i. 持續接收一個 client 的訊息並印出
ii. 如果從接收函式(recv())中回傳值發現 client 已經失去連線)，則跳出持續接收的迴圈，thread 結束，如此一來，就不會有失去連線後還閒置的 thread。
iii. 將接收到的訊息傳給所有 client

Client.c:


設定hint參數
呼叫getaddrinfo尋找可用的structure
遍歷所有structure，並嘗試:
1. 建立socket
2. 連線到指定IP的port 3490
迴圈結束後是否有建立成功
是
否
結束程式
接收使用者名稱
建立接收server的thread
Thread threadrecv
1. while loop持續接收server訊息並印出
2.如果server關閉，關閉程式
while loop持續接收使用者輸入的內容，並傳送給server
使用者是否輸入"quit"
否
是
結束程式

說明:

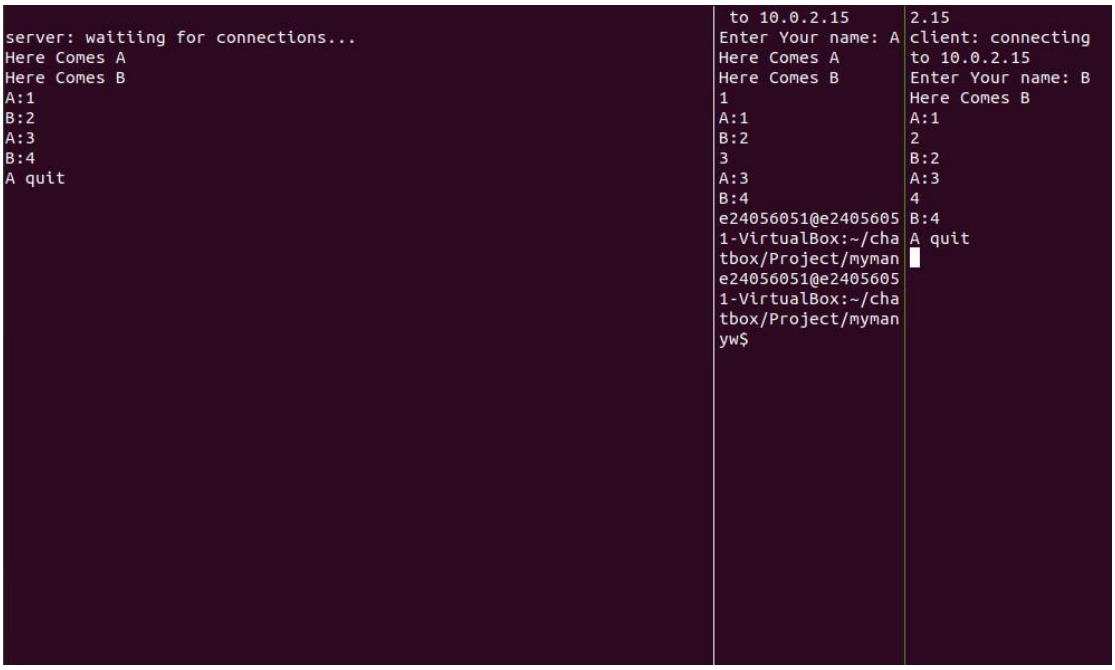
首先一樣利用建立 addrinfo 結構 hint，並設定相關資訊，在此程式中設定了:

i. IP
ii. IP family
iii. TCP stream

接著利用函式 getaddrinfo 尋找符合條件的 structure，並利用 for 迴圈遍歷所有 getaddrinfo 回傳的 linklist，嘗試建立 socket 並連線到 port 3490，如果在遍歷完整個 linklist 後，沒有找到可以使用的 structure，則結束程式。

完成連線後，利用 fgets 接收使用者名稱，並建立接收 server 端的訊息的 thread threadrec，thread 會進入持續接收的 while loop，並利用回傳的值來判斷 server 是否已經關閉。在完成接收訊息的部分後，主程序的部分持續用 fgets 接收使用者的輸入內容，並傳送給 server 端。
如果使用者在輸入內容時，輸入"quit"，則將離開訊息傳給 server，並結束程式。

成果圖:



(二)一對一聊天室

Server.c:

前部分一樣利用 getaddrinfo、listen 建立 socket 與監聽，與多人連線不同的是，在 accept 到 client 的連線後，就不再接收其他連線請求，接著建立兩個 thread，分別負責接收與傳送。

Client.c: 完成與 server 的連線後，建立兩個 thread，分別負責接收與傳送。

● Describe how to run your program and how to use your system.

在 terminal 輸入 make 產生 server.out 與 client.out 輸入./server.out 執行 server 程式

輸入./client.out IP 位址 執行 client 程式(example: ./client.out 10.0.2.15)

對話紀錄將儲存在 database.txt

● Explain the solution of synchronization problems in this project.

在聊天室運作時，可能會發生資料因為沒有及時更新，導致最後 database 錯誤的情況，這種情況產生的原因可能是:

1. A 傳送的資料先被 server 端接收，並開始讀寫 database.txt
2. B 傳送的資料也被 server 端接收，也開始讀寫 database.txt
3. 假設要將 A 資料寫入 database.txt 的是 Server 端的程序 A，要將 B 資料寫入的是程序 B，A 比 B 早且程序 B 要讀寫時，A 還沒完成讀寫，此時 B 會在沒有 A 的新資料的基礎下做更動，導致 A 資料遺失。

為避免這種情況，我們採用了 Semaphore。利用 Semaphore 將全域變數 filesem 資源數設成 1

```
sem_t filesem;
```

```
sem_init(&filesem, 0, 1);
```

並在每次讀寫前都加上 sem_wait(filesem)，完成後都加上 sem_post(filesem)，就可避免同時有兩個程序對 database 做更動。

```
void SendtoAll(char *sentence)
{
    for(int i = 0;i<limit;i++){
        if(usersocks[i] != 0) {
            send(usersocks[i], sentence, strlen(sentence), 0);
        }
    }


    sem_wait(&filesem);
    FILE *in;
    in = fopen("database.txt", "a");
    fprintf(in, "%s", sentence);
    fclose(in);
    sem_post(&filesem);

}
```

I. Mutual Exclusion

當使用 sem_wait 函式時，作為 parameter 的 sem_t 變數會減一，且當 sem_t 小於等於 0 的時候，程序會等待，直到 sem_t 變數大於 0(有資源可以使用)。因此如果我們將資源數設為 1，就可避免兩個程序同時對 database 讀寫。

II. Progress

當 sem_t 變數大於 0 的時候，程序可以對 database 讀寫

III. Bounded Waiting

如果前一個程序一直沒有完成讀寫，其他程序就必須等待到它結束，所以不保證避免 deadlock。

討論:

Socket 與 Select:

如果 Sever 要接收訊息，不斷讀取 Socket 時而發生 blocking 時，select 提供一個可以不用浪費 cpu 而同時又可以監控多個 socket 的功能，哪些 sockets 已經有資料可以讀取、哪些 sockets 已經可以寫入，甚至還可以告訴你哪

些 sockets 觸發了例外，select()的原型如下

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int select(int numfds, fd_set *readfds, fd_set *writefds,
        fd_set *exceptfds, struct timeval *timeout);
```

這個函式以 readfds、writefds 及 exceptfds 分別表示要監視的那一組 file descriptor set（檔案描述符集合），雖然用 select 有相當好的移植性，但在大量連線下的效能會很差。

Socket 與 Libevent:

Libevent 是一個種用 C 語言編寫的輕量級高效能事件通知庫作為底層的網路庫，。編譯庫的程式碼時，編譯的指令碼將會根據 OS 支援的處理事件機制，來編譯相應的程式碼，從而在 libevent 介面上保持一致。

。Libevent 對 socket 裡的 epoll/select 等功能進行了封裝，並且使用了一些設計模式（比如反應堆模式），用事件機制來簡化了 socket 編程。(1)假設有多個 client 同時往 sever 通過 socket 寫 b 入資料，若在此時使用 libevent，server 裡就可以不用 epoll 或是 select 來監控 socket 收到了客戶端來的資料。當某個 socket 裡有可讀數據時，libevent 會自動觸發一個“讀事件”，透過這個 trigger 觸發相對應的函示來讀取 socket 裡面的資料。Libevent 判斷 socket 是否有資料可讀，然後產生相應的程序。(2)對於“寫事件”，libevent 會監控某個 socket 是否可寫，只要可寫，就會觸發“寫事件”，通過 trigger 來調用相應的函數，將數據寫到 socket 裡。