OS 2nd project 多對多聊天室

C24074023 張文哲 github: stupidrara

C24076130 林佑融 github: TooooooT

1. Briefly describe the workflow and architecture of the system.

   Workflow:
   Client 端會執行兩個 process。一個 process 負責讀 user 的訊息處理後傳給 server。另一個 process 負責持續接收 server 傳來的訊息處理後 print 到 terminal，並同時傳送 2 bytes 的 message 以告知 server 此 process 正在連線中。
   Server 端會用 threads 來 handle 來自 client 的兩個 request。接收訊息的 thread 會將 client 傳來的訊息整理後存入 global variable: chatHistory[]中，以 circular queue 來存，過舊的訊息會被新的訊息覆蓋。傳送訊息的 thread 會持續查看 chatHistory 的 rear 有沒有更動，若有更動則傳新訊息給 client。

   Architecture:
   Client 端有兩個 process。一個 process 負責讀 user 的訊息處理後傳給 server。另一個 process 負責持續接收 server 傳來的訊息處理後 print 到 terminal。Server 端會用 threads 來 handle 來自 client 的兩個 request，一個 client 會對應到 server 的兩個 thread。

2. Explain the solution of synchronization problems in this project.

   Mutual Exclusion:
   Server 會將 client 傳來的訊息整理後存入 global variable: chatHistory[]中，此過程會以 mutex lock 保護，防止其他 thread 同時寫入 chatHistory[]，造成資料覆蓋或毀損。

   Progress:
   Server 會將 client 傳來的訊息整理後存入 global variable: chatHistory[]中，此過程會以 mutex lock 保護，沒有其他使用者在修改 chatHistory[] 時，若有其他使用者要修改則一定可以獲取 lock 所以不會造成 deadlock。

   Bounded waiting:
   若 semaphore 的 lock 是隨機讓 client 獲得則不會有 bounded waiting，但

是我們有限制 client 人數且 client 送訊息的速度不會比執行的還快，所以不會有 bounded waiting。

3. Discuss anything as possible:
    1. 一開始寫作業時 Socket 有時會連不上，但重新編譯再執行就可以了，但後來就很少發生這個問題，也不知道原因是什麼。
    2. 之所以 client 不使用 thread 是因為我們使用不同 terminal 視窗處理 I/O，而我們找不到方法讓 thread 來 handle 不同 terminal，因此我們使用 popen()創建新 terminal 並執行 process 在該 terminal，缺點是無法共用 global variable，任何操作都需要透過 server 溝通。

4. How to run:
    1. $ make (on terminal)
    2. $ ./socket_server (execute server)
    3. $ ./socket_client <username> (at most 11 characters) (execute client)

```
$ make
gcc socket_server.c -lpthread -o socket_server
gcc socket_client.c -o socket_client
gcc socket_client_read.c -o socket_client_read
gcc socket_client_chatroom.c -o socket_client_chatroom
```

圖一、步驟一：$ make

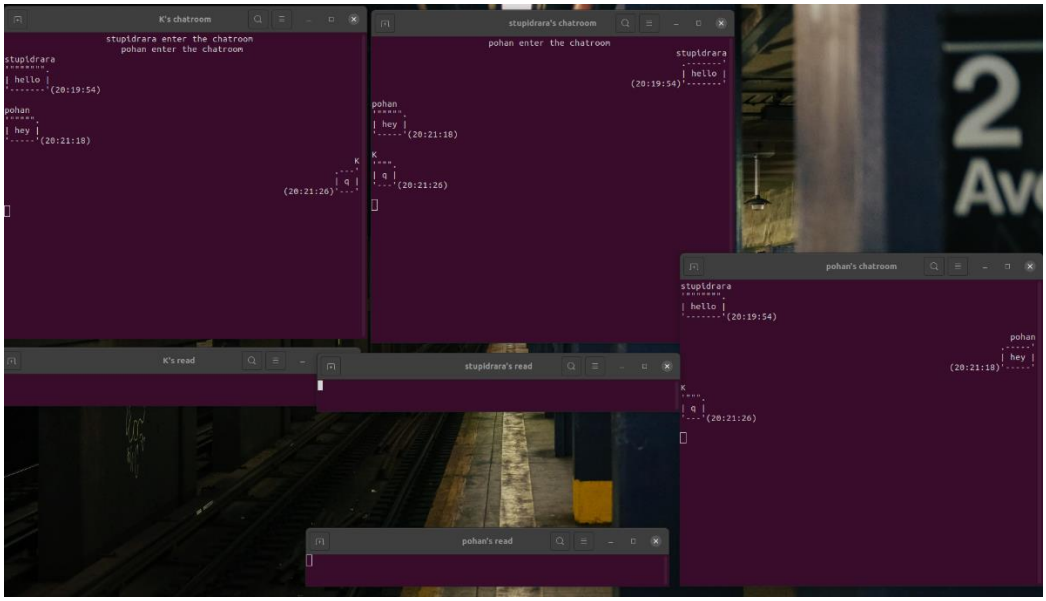
```
$ ./socket_server
```

圖二、步驟二：$ ./socket_server

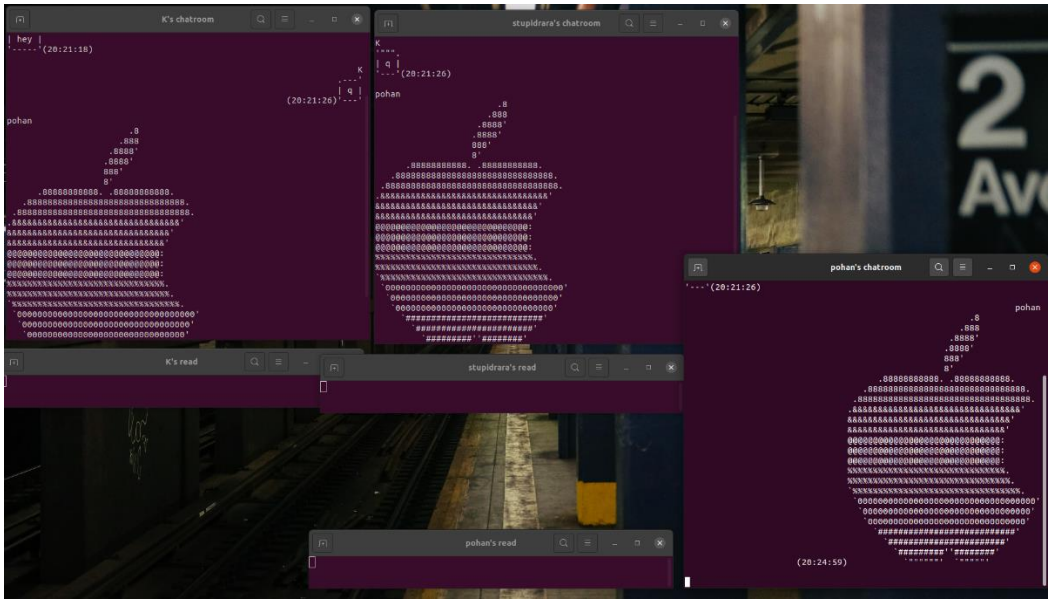
```
$ ./socket_client stupidrara
```

圖三、步驟三：$ ./socket_client <username>

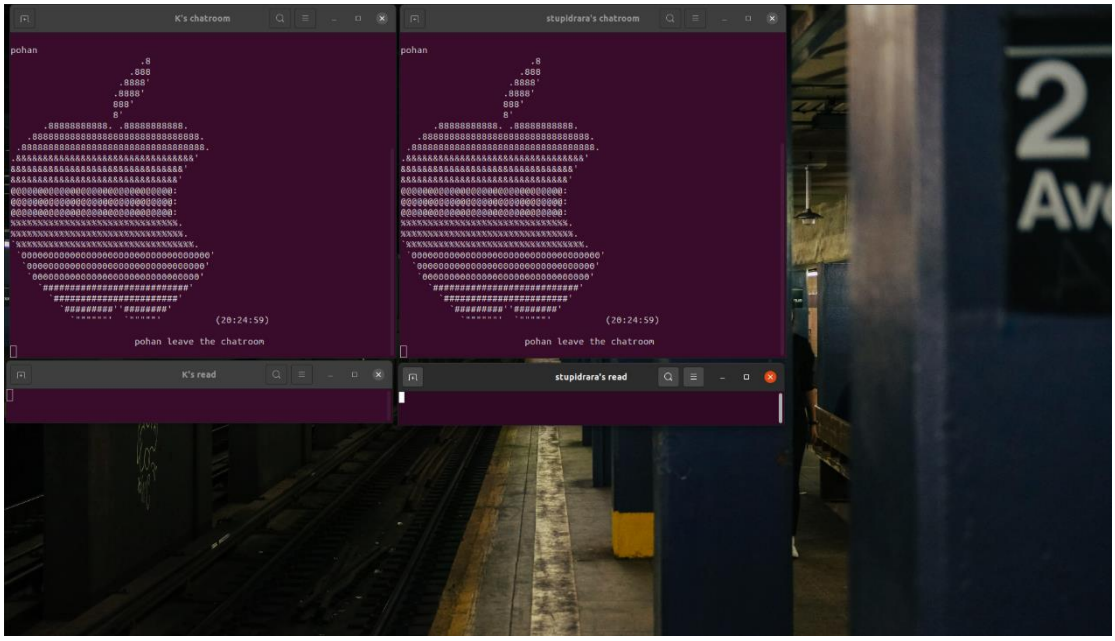
5. How to use:
    1. There will be two window name “username’s chatroom” and ”username’s read”. Read window is for user to write message, chatroom window is for user to see the chatroom message. You should type message in ”username’s read” instead of ”username’s chatroom”.
    2. There are two special command for user to write message.
        1. /quit: quit client program.
        2. /sticker <stickerID> (look sticker.h to see the sticker id): send sticker.
    3. For normal message simply type what you want to send in read window.



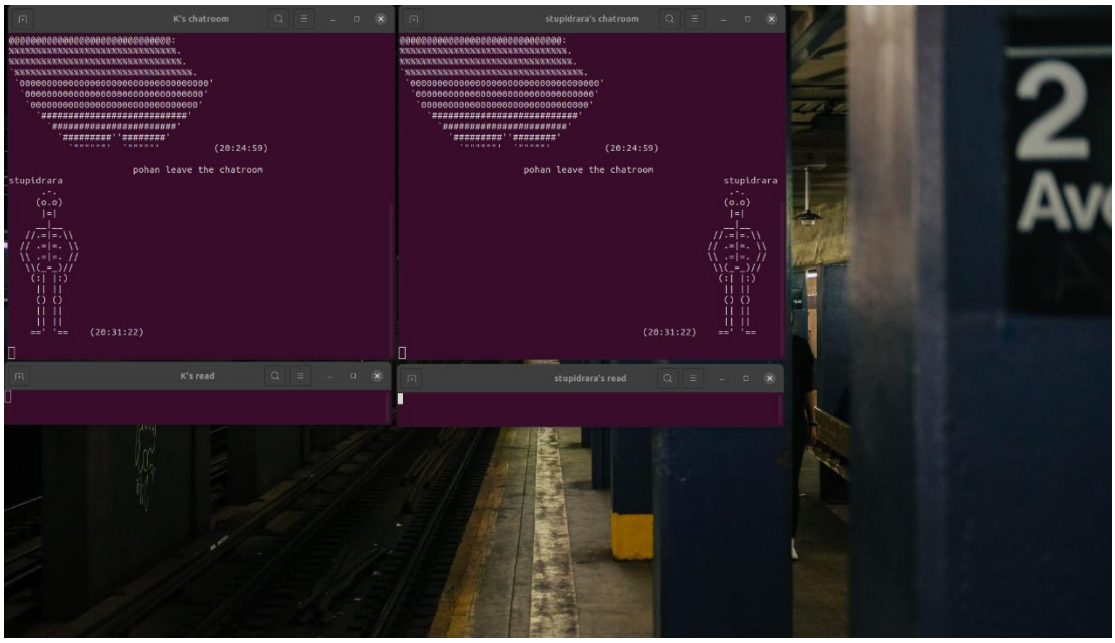
圖四、一般聊天的樣子。可以發現自己講的話會顯示在聊天室右邊。



圖五、pohan 輸入 /sticker 2 的樣子。



圖六、pohan 輸入 /quit 離開聊天室。其他人的聊天室會顯示 pohan 離開。



圖七、stupidrara 輸入 /sticker 8 的樣子。