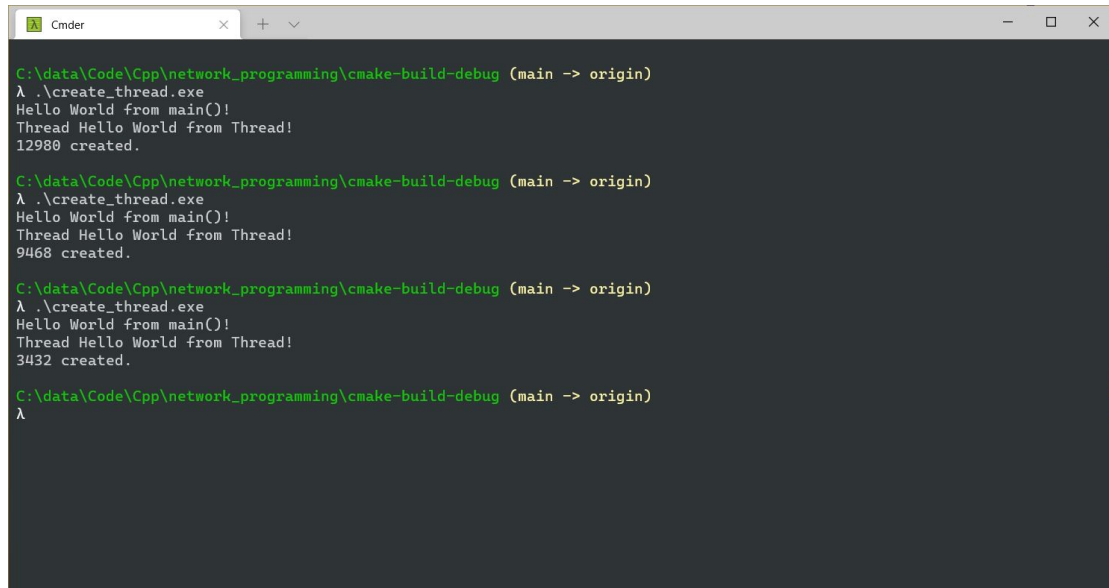


網路程式設計 HW-多緒網路程式作業

一、執行畫面

練習 1：產生 thread

執行三次產生 thread 的程式，可看到每次的 thread id 皆不同。



```
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\create_thread.exe
Hello World from main()!
Thread Hello World from Thread!
12980 created.

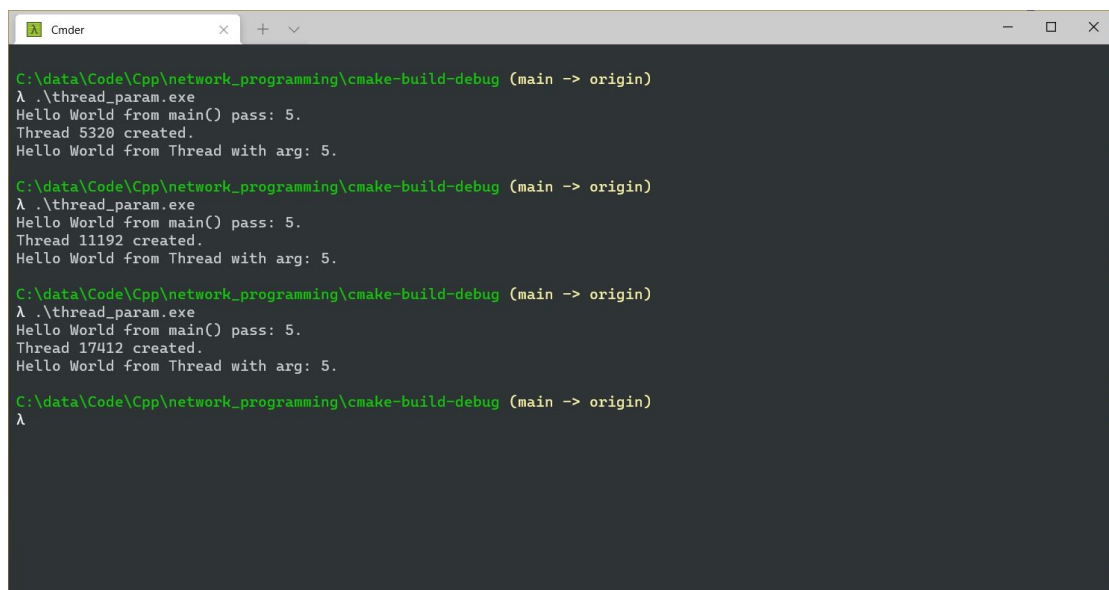
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\create_thread.exe
Hello World from main()!
Thread Hello World from Thread!
9468 created.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\create_thread.exe
Hello World from main()!
Thread Hello World from Thread!
3432 created.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ
```

練習 2：產生 thread 傳遞參數

執行三次產生 thread 的程式，傳遞 int 型態的參數，值為 5，可看到在 thread 中成功接收參數。



```
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\thread_param.exe
Hello World from main() pass: 5.
Thread 5320 created.
Hello World from Thread with arg: 5.

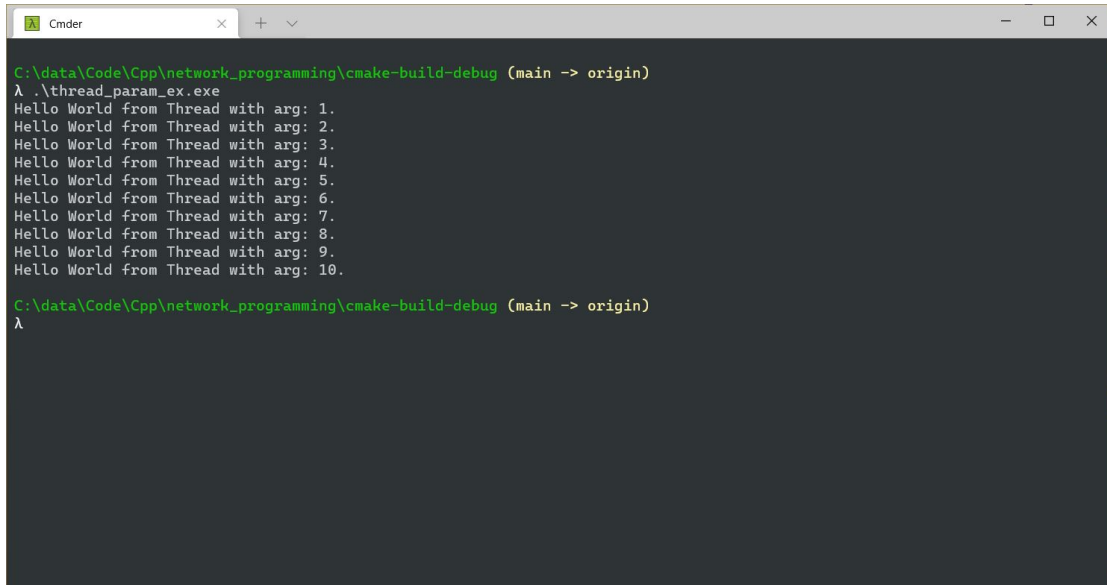
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\thread_param.exe
Hello World from main() pass: 5.
Thread 11192 created.
Hello World from Thread with arg: 5.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\thread_param.exe
Hello World from main() pass: 5.
Thread 17412 created.
Hello World from Thread with arg: 5.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ
```

練習 3：產生 thread 傳遞參數 1~10

使用迴圈產生 10 個 thread，傳遞 int 型態的參數，值分別為 1~10，可看到在 thread 中成功接收參數。



```

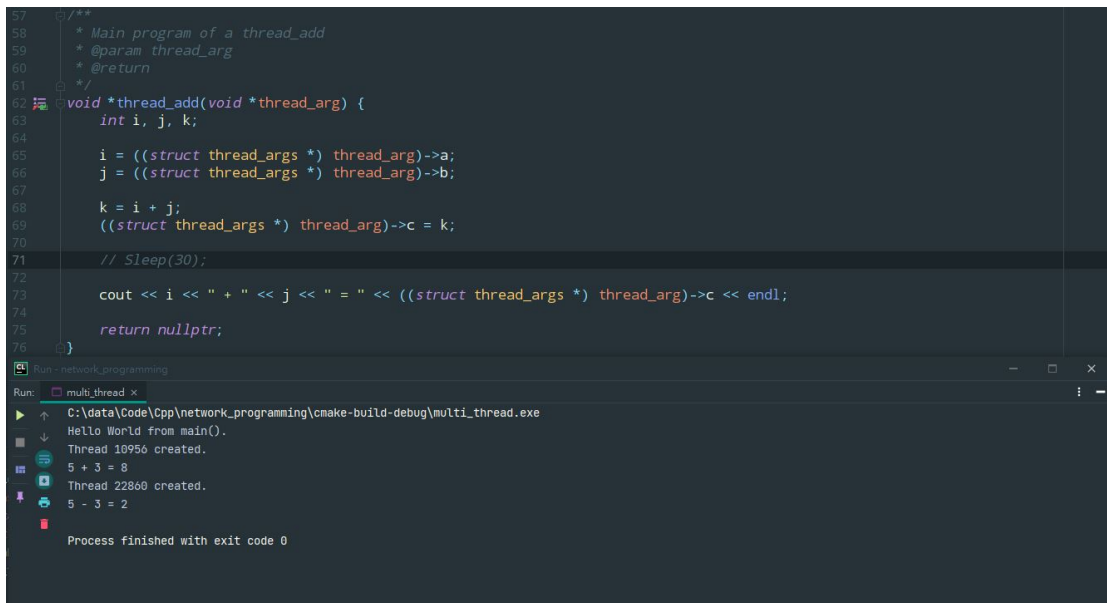
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\thread_param_ex.exe
Hello World from Thread with arg: 1.
Hello World from Thread with arg: 2.
Hello World from Thread with arg: 3.
Hello World from Thread with arg: 4.
Hello World from Thread with arg: 5.
Hello World from Thread with arg: 6.
Hello World from Thread with arg: 7.
Hello World from Thread with arg: 8.
Hello World from Thread with arg: 9.
Hello World from Thread with arg: 10.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ

```

練習 4：Multiple thread 交換資料

產生兩個 thread，分別進行加法和減法的運算。



```

57  /**
58   * Main program of a thread_add
59   * @param thread_arg
60   * @return
61   */
62  void *thread_add(void *thread_arg) {
63      int i, j, k;
64
65      i = ((struct thread_args *) thread_arg)->a;
66      j = ((struct thread_args *) thread_arg)->b;
67
68      k = i + j;
69      ((struct thread_args *) thread_arg)->c = k;
70
71      // Sleep(30);
72
73      cout << i << " + " << j << " = " << ((struct thread_args *) thread_arg)->c << endl;
74
75      return nullptr;
76  }

```

```

Run - network_programming
multi_thread x
C:\data\Code\Cpp\network_programming\cmake-build-debug\multi_thread.exe
Hello World from main().
Thread 10956 created.
5 + 3 = 8
Thread 22860 created.
5 - 3 = 2
Process finished with exit code 0

```

若在建法的 thread 中，運算完後加上 sleep，但是 sleep 的同時，減法也在執行，並且執行完畢後已將 thread_arg 參數 delete 掉，當加法的 thread 再回來輸出時，已無法進行輸出。

```

57  /**
58   * Main program of a thread_add
59   * @param thread_arg
60   * @return
61   */
62  void *thread_add(void *thread_arg) {
63      int i, j, k;
64
65      i = ((struct thread_args *) thread_arg)->a;
66      j = ((struct thread_args *) thread_arg)->b;
67
68      k = i + j;
69      ((struct thread_args *) thread_arg)->c = k;
70
71      Sleep( dwMilliseconds: 30);
72
73      cout << i << " + " << j << " = " << ((struct thread_args *) thread_arg)->c << endl;
74
75      return nullptr;
76  }

```

Run - network_programming

```

Run: multi_thread x
C:\data\Code\Cpp\network_programming\cmake-build-debug\multi_thread.exe
Hello World from main().
Thread 22008 created.
Thread 22008 created.
5 - 3 = 2
5 + 3 = 1769808
Process finished with exit code 0

```

練習 5：Multiple thread 交換資料 1 加到 100

使用迴圈產生 10 個 thread，傳遞 int 型態的參數，可看到在 thread 中成功接收參數，並分別運算每 10 個數字的總和，回傳至 total 變數中。

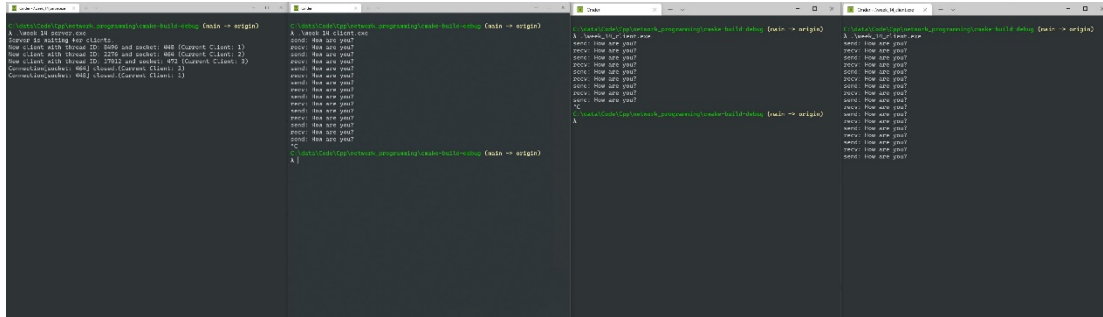
```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\multi_thread_ex.exe
Sum of 1 ~ 10 is 55, total is 55
Sum of 11 ~ 20 is 155, total is 210
Sum of 21 ~ 30 is 255, total is 465
Sum of 31 ~ 40 is 355, total is 820
Sum of 41 ~ 50 is 455, total is 1275
Sum of 51 ~ 60 is 555, total is 1830
Sum of 61 ~ 70 is 655, total is 2485
Sum of 71 ~ 80 is 755, total is 3240
Sum of 81 ~ 90 is 855, total is 4095
Sum of 91 ~ 100 is 955, total is 5050
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ

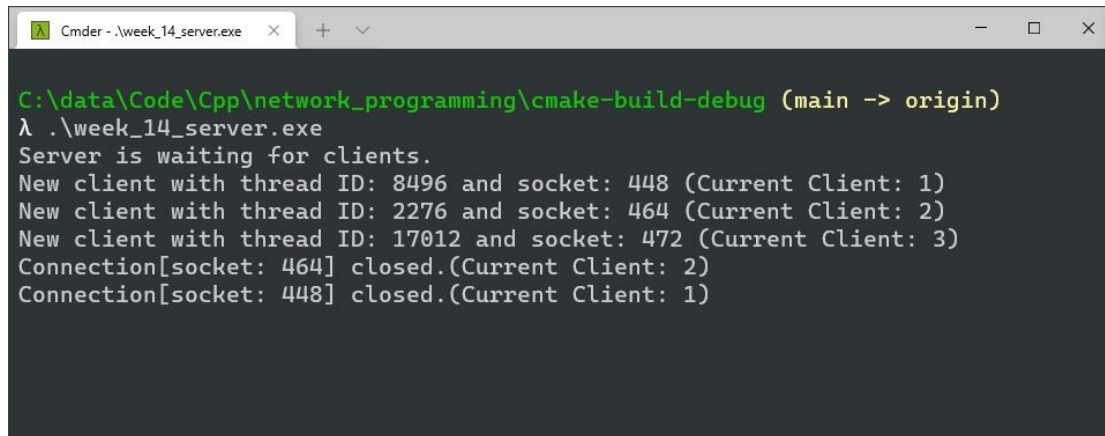
```

進階練習 1：Echo Server 多工

下圖最左邊的視窗為 server 的畫面，其他視窗為 client 的畫面，server 可對於每個 client 建立一個 thread 來服務，即使臨時有 client 加入或離開，server 皆可運作，更可知是哪一個 client 連入及離開。



下圖為 server 端的畫面，可看到 client 1、client 2 及 client 3 依序連入，但是 client 2 中途離開，之後 client 1 也中途離開，只剩 client 3 仍在連線。

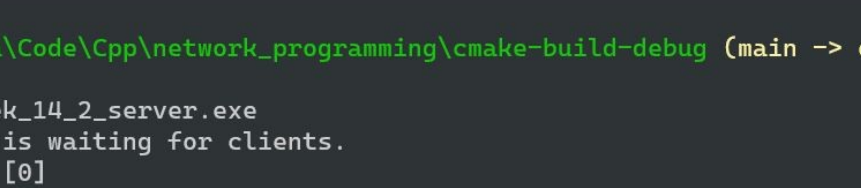


進階練習 2：聊天室 Server 多工(Non-blocking)

下圖最左邊的視窗為 server 的端畫面，中間及右邊的視窗為 client 端的畫面，利用 client 建立 2 個 socket 來連線到 server，模擬兩個使用者在聊天，可看到此版的聊天室 server 可同時支援多個使用者同時連入來聊天，而且同時支援開啟多個聊天室。

[illegible]

下圖為 server 端的畫面，可看到 client 1 的兩個 socket 連入後，client 2 的兩個 socket 也連入了，但是 client 1 中途離開，只剩 client 2 仍在連線，仍可繼續維持服務。



The screenshot shows a Windows Command Prompt window with the title bar "Cmder - .\week_14_2_server.exe". The command prompt is running a C++ program located at "C:\data\Code\Cpp\network_programming\cmake-build-debug". The program's output is as follows:

```
C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)

λ .\week_14_2_server.exe
Server is waiting for clients.
accept [0]
accept [1]
New chat room with thread ID: 3596 and sockets: 444 and 448
Server is waiting for clients.
accept [0]
accept [1]
New chat room with thread ID: 22436 and sockets: 464 and 424
Server is waiting for clients.
Disconnected! error code: 10054
```

進階練習 3：聊天室 Client 多工

下圖最左邊的視窗為 server 的畫面，中間的視窗為 client 1 的畫面，右方的視窗為 client 2 的畫面，可看到 client 1 及 client 2 在聊天室中可同時輸入並接收訊息，此處截圖為單一聊天室，兩間聊天室的截圖請參閱心得的部分。

The image shows three terminal windows side-by-side. The left window is the server process, showing it listening on port 4444 and accepting connections. The middle and right windows are client processes, showing a sequence of send and receive messages between them.

```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_server.exe
Server is waiting for clients.
accept [0]
accept [1]
New chat room with thread ID: 5689 and sockets: 448 and 444
Server is waiting for clients.

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_client.exe
abc
send: abc
recv: def
ghiii
send: ghiii
recv: hihi
recv: hello
recv: ouch
hahaha
send: hahaha

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_client.exe
recv: abc
def
send: def
recv: ghiii
hihi
send: hihi
hello
send: hello
ouch
send: ouch
recv: hahaha

```

下圖的視窗為 client 1 的畫面。

The image shows a single terminal window titled 'Cmder - .\week_14_3_client.exe'. It displays the same sequence of send and receive messages as the client 1 window in the previous image.

```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_client.exe
abc
send: abc
recv: def
ghiii
send: ghiii
recv: hihi
recv: hello
recv: ouch
hahaha
send: hahaha

```

下圖的視窗為 client 2 的畫面。

The image shows a single terminal window titled 'Cmder - .\week_14_3_client.exe'. It displays the same sequence of send and receive messages as the client 2 window in the previous image.

```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_client.exe
recv: abc
def
send: def
recv: ghiii
hihi
send: hihi
hello
send: hello
ouch
send: ouch
recv: hahaha

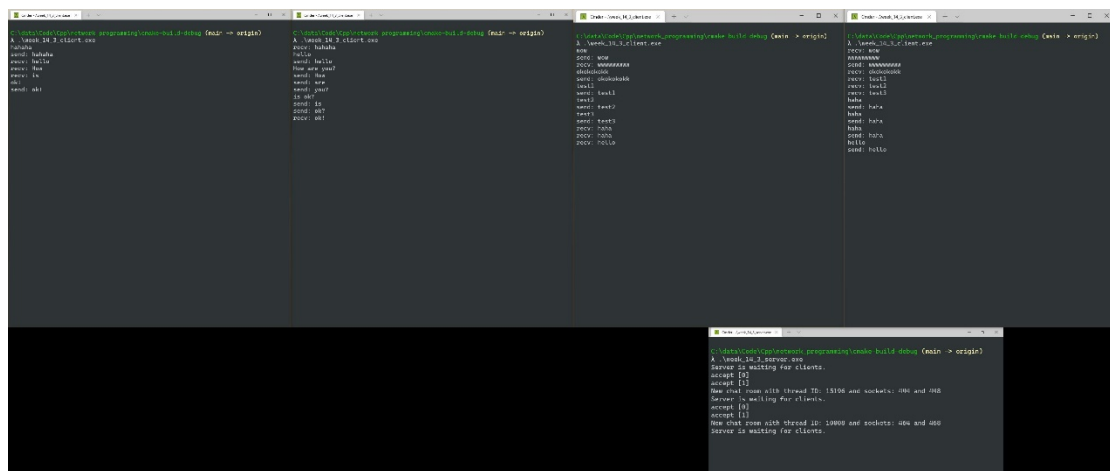
```

二、心得

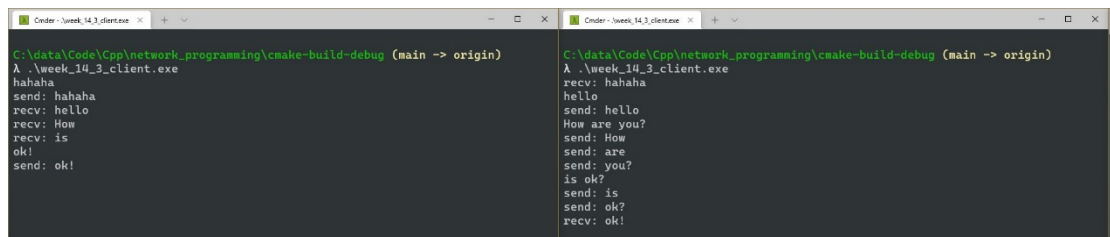
這次作業終於來到了多執行緒的部分，但我覺得大二的同學尚未學過作業系統，如果沒有自己研究一番的話，即使寫完程式後，可能也不太清楚它的作用。

這次因為遠距教學的緣故，我花了一個下午到晚上來完成這次的作業，我從 0 開始重寫，複習之前的程式碼，雖然花了不少時間(大概有將近 2 倍的課堂時間這麼多)在實作，但真的值得。

在實作進階練習題第3題時，我發現了一些問題，如下圖所示，若以下圖片不清楚，下一段有分開的截圖。



在傳送時偶爾會發生傳送無法順利接收到的情況，像是下圖傳送了 How are you? 這個句子，看 client 傳送的情況是分成 How 和 are 和 you? 來傳送出去，照理來說應該會收到 3 個部分，卻只接收到 How 的部分，真的是有一些問題。



這是另一間聊天室傳送的情況，連續傳送了 3 個 haha，卻只接收到了兩個。

```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_client.exe
wow
send: wow
recv: wow
send: okokokokk
recv: okokokokk
send: test1
recv: test1
send: test2
recv: test2
send: test3
recv: test3
send: haha
recv: haha
send: haha
recv: haha
send: hello
recv: hello
  
```

而這個是 server 端的運作情況，看似蠻正常的。

```

C:\data\Code\Cpp\network_programming\cmake-build-debug (main -> origin)
λ .\week_14_3_server.exe
Server is waiting for clients.
accept [0]
accept [1]
New chat room with thread ID: 15196 and sockets: 444 and 448
Server is waiting for clients.
accept [0]
accept [1]
New chat room with thread ID: 10008 and sockets: 464 and 468
Server is waiting for clients.
  
```

不過因為時間的關係，我就沒有再詳細的去研究事發的原因了，感覺 Windows 作業系統中出這些狀況也並不意外，實際在撰寫程式時多少會加上 error detection 和 checksum 的機制，應該是可以解決這個情況所發生的問題。

寫著寫著，看到我這次的作業已經編號到第 9 號了，其實我很想聽老師講 Unix socket 的部分，畢竟對我來說，最實用的是在 Linux 系統上實作 socket，因為我很少使用到 Windows 作業系統，每星期除了星期一上網路程式設計的課程以外，幾乎不會使用到 Windows，所以真的很希望老師除了 Windows sock 以外，多教教其他的部分，像是 Python socket 之類的都很棒。