MP4　　B04705018　資管二　楊之郡

Thread pool 的 implementation:
我在 csiebox_server.c 裡把 mutex、condition variable 跟自己定義的 struct threadpool 宣告成 global variable，並且在 server_run 的地方依照 server 的 config 裡 create 一樣數量的 thread，讓他們去執行 handle_request，handle_request 中先用搶 mutex，搶到的 thread 執行 pthread_cond_wait 等待 server 將 client 的 request 分配進來，分配到 request 的 thread，先將 conn_fd 從 select 中移除，避免重複接收，之後等到 request 結束後再加回 select，而當 thread 被 assign 一個 request 時，先搶 mutex 避免衝突，將 struct threadpool 裡的 count 加 1，記錄目前正在運作的 thread 的數量。在被叫醒的 handle_request 中，先搶 mutex，將他連線的 client 的 conn_fd 複製一份，unlock mutex，因為 thread 的參數是放在同一個變數裡。thread 結束時，一樣先搶 mutex，然後將 count 減 1，server 在讀 request 時先檢查 count 跟 threadnum，如果 count 等於 threadnum 代表沒有足夠的 thread 可以使用，server 就回傳 busy 給 client。

如何用 process 去 implement:
一有 client 的 request 進來就 fork 一次 process，讓 child process 去處理 client 的 request，處理完後就呼叫_exit()去 terminate child process，parent process 繼續接收其他 client 傳進來的 request。

Multi-thread 和 Multi-process 比較:

1. Thread 相對於 process 比較好去 start 跟 terminate，因為要 fork 一個 process 必須要把所有的東西都複製一份，而 thread 的 memory 是 shared，所以 process 在記憶體的使用上會比 thread 多很多，因為 fork 會把其他在 handle_request 時不需要用到的東西也複製了一份。
2. CPU 在 Task-switch 時 thread 比較快，而在 process 之間切換會比較慢。
3. Thread 因為 shared data 的關係，需要使用 mutex 避免產生 race condition，而且假如一個 thread 出問題了話，其他 thread 也會受到影響，process 因為 memory 是分開的所以不用考慮這個問題。
4. Thread 在 debug 方面會比較困難，因為是 shared memory，一個 thread 有問題的原因可能是其他 thread 產生錯誤而造成，process 在 debug 上就較為容易。