-> Description of thread pool implementation

我用一個structure存使用thread pool需要用到的大部份東西，像是thread pool 中的每個 thread，我用一個型態為pthread_t的指標當成陣列頭來存，並用一個 thread_count 紀錄目前有多少thread正在執行任務，用以判斷是否busy的狀況，以及使用一個變數todo_fd紀錄這個即將要被處理的連線，讓thread pool其下各個thread去搶它來做，和使用一個dealfd陣列去記錄現在正在處理的conn_fd，避免兩個thread同時處理到同一個conn_fd的要求。並存取 pthread_cond_t 以及 pthread_mutex_t 讓thread可以等在pthread_cond_t上以及讓 thread pool可以上鎖，暫時不受其他thread影響

thread pool 的 structure：

```
/* typedef struct Threadpool_t {
        pthread_mutex_t lock;
        pthread_cond_t notify;
        pthread_t *threads;
        int thread_count;
        int todo_fd;
        int dealfd[20];
    } threadpool_t; */
```

而再來使用pthread_create 創出server要求的thread 數量：

```
for( i=0 to N ) pthread_create(&(pool->threads[i]), NULL, threadpool_thread, (void *)
(*server));
```

並把原先在server_run的handle_request（）放進 thread要執行的 threadpool_thread（）函式交由thread去實作

-> Discuss how to use process instead of thread to handle multiple clients and compare throughput of these two approaches (for example memory consumption and delay time)

1.how to use process instead of thread to handle multiple clients：

將整個要處理client要求的程式碼都放進子進程中，包括所有會用到的變數，將子進程當成獨立於父進程外的另一個程式，交由其執行client所送達的要求 。

2. compare throughput of these two approaches ：

(1) memory consumption: process > thread
process is a collection of code, memory, data and other resources.
A thread is a sequence of code that is executed within the scope of the process.and the threads of a process share the same memory while the processes aren't.

(2) delay time: process > thread
thread在程式一部分被暫停或是執行操作時間冗長時仍可以繼續執行而process則需要等待cpu的排程

(3) Threads can directly communicate with other threads of its process; while processes must use interprocess communication to communicate with their sibling processes

(4) New threads are easily created; new processes require duplication of the parent process

(5) Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes.

(6) Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process do not affect child processes.