Paper Report　b01209041 潘煒凱

1.
使用作業一的 LCS trace，如同 diff 程式，在多個 client 共同修改的地方加入 conflict area，紀錄 client 各自修改的內容，其他未修改的地方保留。
或是當一個 client 在修改檔案上傳的時候，server 就要將該檔案 lock 住。

2.
當 server 收到一個 request 後，就 fork 一個 process 來 handle request，直到處理完後 terminate。

需要限定可以 fork 的 process 數量，因此會用到 exit 以及 wait，當 process 使用數量達上限，request 就要進入 queue，或是像作業一樣給 client message，並要求再傳 request。
因為有 exit，所以要處理 zombie process，deadlock，race condition 的情形。

3.
使用 fork：
server 對每一個連上線的 client 會 copy 一整份 process 的記憶體，因此每個處理 client 的 process 都有自己的變數，各 process 間不會互相干擾，不會一個處理一個 client 的 server terminates，整個 server 就壞掉。
但是 process 間的 shared memory 表現不好。以及 process 間 context switch 也會對效能，記憶體使用有所影響。

使用 thread：
server 在自己的 process 中，對每一個連上線的 client 各自開 thread stack。
Thread 間會共用 global 跟 heap 中的變數，因此 shared memory 表現較好，但是有可能產生 race condition，因此 thread 間的溝通很重要，必要時需要使用 mutex_lock，來避免 race condition 發生。
由於 thread 都是在同一個 process 裡面，就不需要有 context switch，start up time 也快，記憶體用量也不像 fork 那麼大量。只是有可能一個 thread terminates 的話，整個 process 也會 terminates。