

甲

題目一：執行 1 分鐘以後結束執行，並印出以 reader 和 writer 的身份，進入 critical section 的次數。

```
概覽 終端機 12月2日 22:59 s407210013@osdl2: ~/osdl/sharedFolder/hw8
total number of writer entering CS = 1414/sec
parallel level of read = 10.153678
nTicket = -15
total number of reader entering CS = 185022/sec
total number of writer entering CS = 1581/sec
parallel level of read = 10.158679
nTicket = -1010
total number of reader entering CS = 189246/sec
total number of writer entering CS = 2005/sec
parallel level of read = 10.151606
nTicket = -16
total number of reader entering CS = 161967/sec
total number of writer entering CS = 850/sec
parallel level of read = 10.156423
nTicket = -1016
total number of reader entering CS = 183170/sec
total number of writer entering CS = 1348/sec
parallel level of read = 10.164164
nTicket = -6
total number of reader entering CS = 188457/sec
total number of writer entering CS = 1863/sec
parallel level of read = 10.157537
nTicket = -2
total number of reader entering CS = 197303/sec
total number of writer entering CS = 2368/sec
parallel level of read = 10.143379
nTicket = -1014
total number of reader entering CS = 170918/sec
total number of writer entering CS = 1818/sec
parallel level of read = 10.127114
nTicket = 995
total number of reader entering CS = 184796/sec
total number of writer entering CS = 2209/sec
parallel level of read = 10.110456
經過60秒...
totally_reader = 10936949
totally_writer = 93105
s407210013@osdl2:~/osdl/sharedFolder/hw8$
```

題目二：執行 1 分鐘後結束，計算加入 memory\_order，reader 和 writer 進入 critical section 次數。

```
概覽 終端機 12月2日 23:02 s407210013@osdl2: ~/osdl/sharedFolder/hw8
total number of writer entering CS = 769/sec
parallel level of read = 10.923929
nTicket = 995
total number of reader entering CS = 178855/sec
total number of writer entering CS = 899/sec
parallel level of read = 10.926208
nTicket = -32
total number of reader entering CS = 178503/sec
total number of writer entering CS = 766/sec
parallel level of read = 10.936449
nTicket = -16
total number of reader entering CS = 158486/sec
total number of writer entering CS = 778/sec
parallel level of read = 10.929046
nTicket = -1017
total number of reader entering CS = 182859/sec
total number of writer entering CS = 1116/sec
parallel level of read = 10.929064
nTicket = 995
total number of reader entering CS = 178306/sec
total number of writer entering CS = 1127/sec
parallel level of read = 10.933053
nTicket = -13
total number of reader entering CS = 178710/sec
total number of writer entering CS = 777/sec
parallel level of read = 10.941409
nTicket = -12
total number of reader entering CS = 174593/sec
total number of writer entering CS = 719/sec
parallel level of read = 10.946073
nTicket = -1020
total number of reader entering CS = 177006/sec
total number of writer entering CS = 688/sec
parallel level of read = 10.956403
經過60秒...
totally_reader = 10681774
totally_writer = 60211
s407210013@osdl2:~/osdl/sharedFolder/hw8$
```

乙

使用 atomic 相關的函數，且「指定 memory\_order」，像是

`atomic_store_explicit(&rd_in_cs_success, 0, memory_order_relaxed)`

，實測出來的結果會是原本進入的次數較多。感覺上因為 `memory_order_seq_cst` 為其預設值，跟後來使用的 `memory_order_relaxed` 相比，後者對於資料的順序不加以限制，而前者對於這部份有較強的處理，因此有結果的產生。假設今天有更強的 `memory_order`，相較此結果，或許就能增加實驗過程中的次數，由此可知可透過 `memory_order` 增加效率。

概覽 終端機 12月2日 23:38 英 s407210013@osdl2: ~/hw8

```
37
38 totally_reader += rd_in_cs_success;
39 totally_writer += wrt_in_cs_success;
40 times++;
41 if(times == 10){
42     printf("經過%d秒...\n", times);
43     printf("totally_reader = %lld\n", totally_reader);
44     printf("totally_writer = %lld\n", totally_writer);
45     keepgoing = 1;
46 }
47
48 atomic_store_explicit(&rd_in_cs_success,0,memory_order_relaxed);
49 atomic_store_explicit(&wrt_in_cs_success,0,memory_order_relaxed);
50 alarm(1);
51 }
52
53 void init_rwlock() {
54     atomic_store_explicit(&nTicket, maxTicket,memory_order_relaxed);
55 }
56
57 void wrt_lock() {
58
59     while(1) {
60         atomic_fetch_sub_explicit(&nTicket, maxTicket,memory_order_relaxed);
61         int ret=atomic_load_explicit(&nTicket,memory_order_relaxed);
62         if (ret == 0) { //success
63             //atomic_fetch_add(&wrt_in_cs_success, 1); //for debugging
64             return;
65         }
66         else {
67             atomic_fetch_add_explicit(&nTicket, maxTicket,memory_order_relaxed);
68         }
69     }
70 }
71
72 void wrt_unlock() {
```

[/home/normalUsers/s407210013/hw8/rwlock\_wo\_order.c][utf-8, unix, c][4KB] 1,48/158 [ 30%]