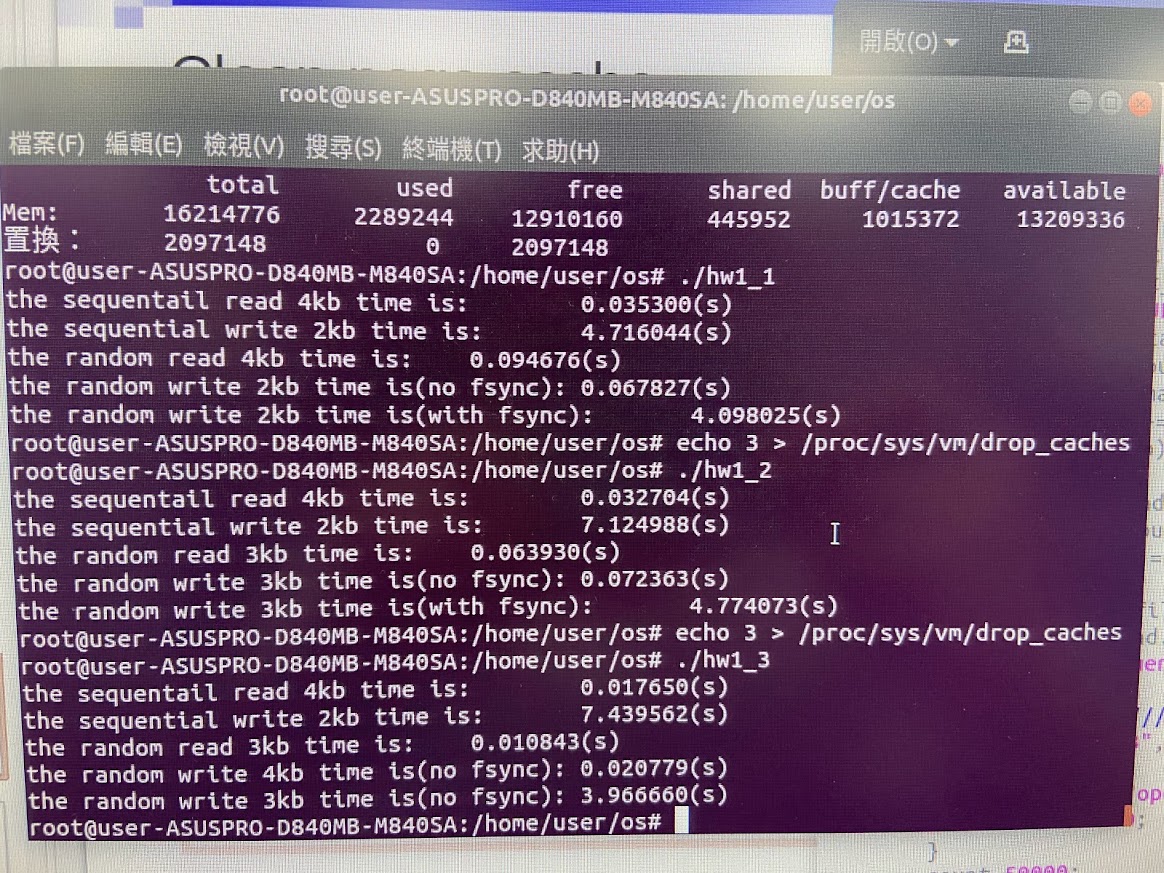
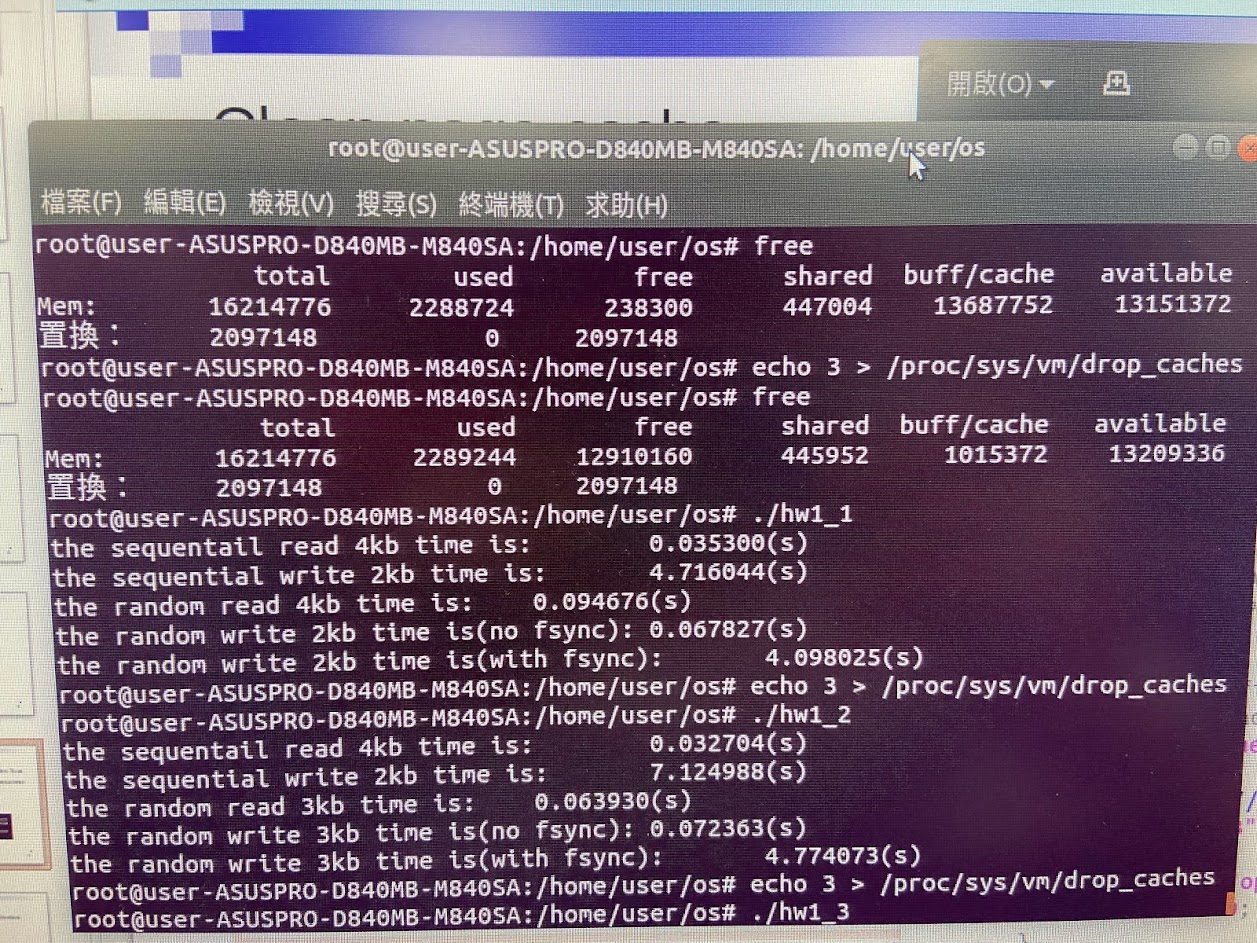
資工三 4108056004 王思正



1\_4

1.(外部function)

Sequential read 4k: 0.035300

Sequential write 2k: 4.716044

Random read 4k: 0.094676

Random write 2k: 0.067827

Random write with fsync() 2k: 4.098025

(a)因為使用SSD所以sequential 與 random 時間不會差很多

(b)在random write時若沒有fsync() 確保寫入就會快非常多，反之就明顯慢了下來`；因為寫入時一次只寫入一點，與硬碟的設計概念有所出入(硬碟的優點就在於累積到很多再一次全部寫入)，這也是變慢的原因

(c)照理來說random write會比sequential write慢，仔細想後才想起random只有做50000次，而sequential做了200M/2K=100K=1024000次，所以random比較快是合理的

2.(system call)

Sequential read 4k: 0.032704

Sequential write 2k: 7.129488

Random read 3k: 0.063930

Random write 3k: 0.072363

Random write with fsync() 3k: 4.774073

(a)因為略過了library直接使用system call，照理來說要比較快才對，可以看到sequential read, random read都比較快

(b)而write所有部分system call比function call情況還要慢，Library自己管理一塊buffer(256~8192 bytes)，fwrite時會先copy到buffer內而非直接呼叫write()，等buffer到滿時才呼叫write() system call 寫到OS page cache，所以就算我們設一次寫1byte，也不會每一次都呼叫write()，大幅的減少system call overhead，因此fwrite()比直接write()快是正常的

(c)再繼續延伸問題，所以我們得到結論:

在單次寫入size小的時候，fwrite會比write表現好=>因為少了許多system call overhead

在單次寫入size大的時候，write會比fwrite表現好=>因為fwrite需要COPY，所以單次寫入size愈大，效能就被write追過

3.(memory-map)

Sequential read 4k: 0.017650

Sequential write 2k: 7.439562

Random read 3k: 0.010843

Random write 4k: 0.020779

Random write with fsync() 3k: 3.966660

(a)跟前面相比又更快了，因為略過了Library以及OS直接對Memory進行處理