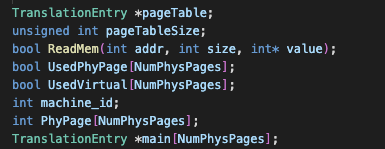
作業系統 Report

作業一 R10945061 林宇恆

# Motivation:

根據作業要求敘述，需要各自執行 matmult 以及 sort 兩個測試檔，且需要修改 sort.c 來達到相同的結果，而以上兩個檔案都需要大量記憶體來運行，因此根據作業要求concurrently，也就需要上次作業的multithread來協助達到共行。根據作業要求使用pageTable，FrameTable以及SwapTable來解決有限記憶體的問題，分別代表virtual，physical跟swap，因此如下圖所示，在machine.h中定義以下表格來記錄使用過程。

發想為透過pageTable將logical memory映射到physical memory，映射後pageTable透過1bit的標記判斷page有沒有效，若有效則代表該page還在main memory中執行，無效則代表page在輔助記憶體中（如硬碟,SSD）。

根據作業提示在以下file中執行。



# Implementation:

# 在userkernel.h中建立一個新的SynchDisk(SwapDisk)，且初始化來模擬輔助記憶體。

# 

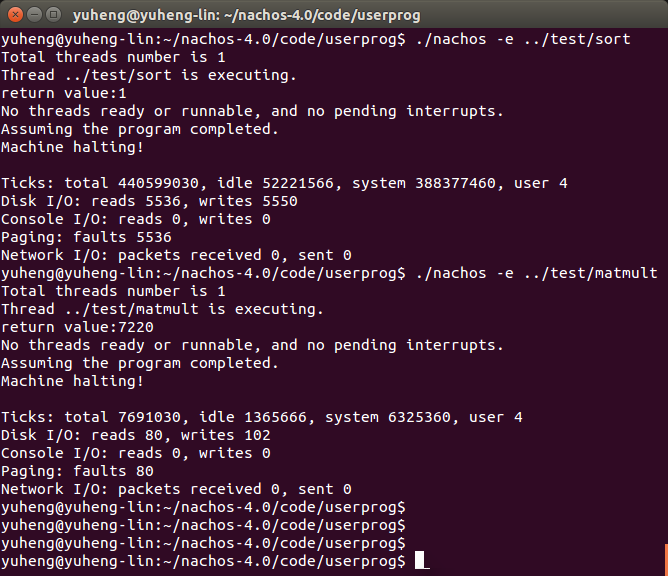
# 在addrspace的load函數中新增，用for迴圈來尋找可用空間，j則用來檢查第幾個frame已被使用，若使用則加一，若j小於NumPhysPages，也就是還有空frame時，就可以將Page放入main memory，第二種情況是main memory滿了，則透過下面的while迴圈檢查virtual memory，最後將其放入輔助記憶體，達到context-switch。

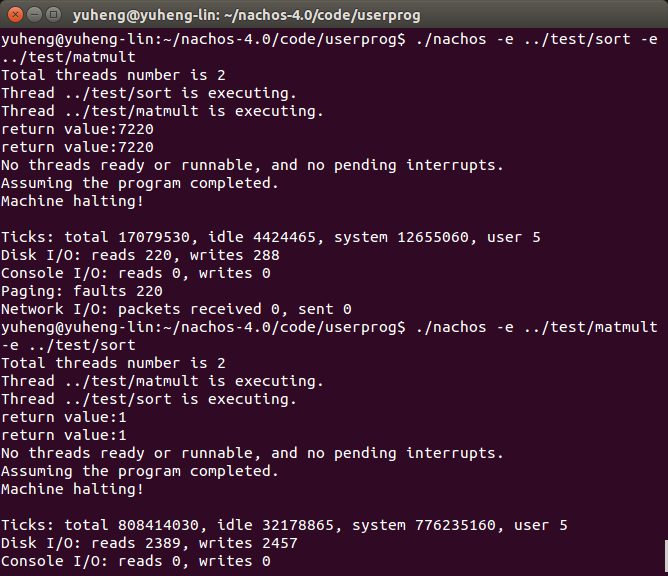
# 演算法的部分使用了Random replacement algorithm，他是隨機選擇一項且在必要時丟棄，因此他不需要保留有關訪問歷史的任何資訊。第一個if條件為當main memory沒滿的情況下載入page到main memory中，buffer則用來暫時儲存page，第二格條件則為當main memory滿了的情況，dead則為被選到的人因為之後就會丟棄因此必死，32為32位元。

# 

# Result:

# 分開執行的結果如下，各自都可以達到準確的答案。



但一開始沒有使用multithread導致執行結果如下圖，共同執行的話結果會跟第二個測試檔相同。

最後加入了multithread後就能夠得到正確答案了。