學號:B06902136
系級:資工三
姓名:賴冠毓

1.設計

(1)kernel_files:

get_time.c: 獲得時間資訊。

show_time.c: 將時間的相關資訊秀在 dmesg。

(2)main.c: 讀取 input 並記錄相關數據，判斷完 policy 後丟到 scheduling 去做排程。另外記下一單位時間的函數。

(3)function.h: 標頭檔，存放所有 define, structure, global variables 跟 function prototype。

(4)process.c: 記錄所有與 process 相關的 function 用法

assign_CPU: 指派 process 到特定的 CPU 上。

wake_up_process: 把在 ready queue 裡的 process 喚醒來執行。

create_process: 在輪到 process 要執行時 fork 產生它，在裡面實際跑執行所需要的時間，再透過自己所寫好的 system call 讓他可以在 dmesg 上秀出資訊。

block_process: 把當前的 process 弄到 waiting queue 裡 block 住。

(5)scheduling.c:

scheduling: 先把 main process(處理 schedule 的這個 process)指派到特定 CPU 上，避免它會 interrupt 其他我們所需要生成的 process。再來將所有 process 的 pid 先設成 0(表示還沒有被產生)，並且根據 process 的 ready time 去做排序，方便之後操作。接下來來開始處理排程，先檢查目前是否有已經完成的 process，如果有的話等待它結束並將跑完的 process 數量加一；再來檢查目前是否有 process 需要被產生，如果有將它創造出來並且先 block 住它，等待時機執行；之後挑選接下來要執行的 process，如果接下來要執行的 process 與目前執行的不同，則進行 context switch；最後將單位時間加一並繼續下一輪。

pick_next_process: 先檢查是否是 non-preemptive policy，如果是的話不可以在當前 process 執行完前換別的 process 執行。其他的就按照個別 policy 判斷。

pick_by_FIFO: 挑選 ready time 最小的 process 執行(最早進來)。

pick_by_RR: 如果目前沒有正在跑的 process，就挑 index 最小的；如果時間剛好到 500 的話，表示需要換下一個 process 執行了；其他狀況則代表時間未到，繼續跑原本的 process。

pick_by_SJF_or_PSJF: 挑選剩餘時間最小的 process 執行。

2.核心版本
Linux-4.14.25(HW1 中所下載的)

3.比較實際結果與理論結果，並解釋造成差異的原因
實際結果與理論結果中，在 stdout 檔案中 process 完成順序是一樣的，但是透過 dmesg 可以發現，當有 process 同時產生搶 CPU 時，kernel 排程的順序未必完全按照理論結果，但若無同時產生 process 則沒有此問題。另外實際執行完 process 的時間也不一定照理論結果，這可能是 CPU 在 context switch 下的延遲所導致。