Operating System Project1 Report

系級：電機四　學號：B05202061　姓名：陳威旭

1. 設計：
   A. Kernel file : 總共有兩個新的 system call 333, 334
      I. get time(333)：使用 getnsitimeofday，將 kernel time 回傳給 user 做紀錄。
      II. print dmesg(334)：傳入 start time, end time，並用 printk 把資訊印出來。

   B. Process block : 主要管理 process 用
      I. struct Process：存有 process 的 ready time, remain execute time, pid，且把讀進來的 input process data，分別放入各個 process block 內。
      II. assign core：將 scheduler process 放到 CPU2，child process 放到 CPU3。
      III. process kick：利用 sched_setscheduler，將 process 的 priority 設定最低。
      IV. process highest：利用 sched_setscheduler，將 process 的 priority 設定最高。
      V. process create：傳入 process execute time 並創造出 child process，使其直接開始一直 run time unit。

   C. scheduler：四大 method
      I. FIFO：由於助教說測資 start time 確定已排好，故我就直接依照順序，create process -> run -> waitpid，直到全部跑完。
      II. SJF：每次從已 start 的 process 中，選出 remain execute time 最短的，給他最高 priority 去跑，直到跑完，再去選下一個，或 stall 住，等到下一個可選 process 出現。
      III. PSJF：每當有任一個新 process 的 start time 到了，就暫停動作去判斷目前全部 process 中哪個 remain execute time 最短，讓他開始(繼續)跑，一直跑直到他跑完，或者被另一新 ready process 中斷。
      IV. RR：用一個 for 迴圈，一直依序確認 process 1~N，看誰想要跑，若 i 想跑，就讓他跑 500 Time unit，或跑完了提早結束，時間到了或結束後，就繼續確認 i+1 是否要做。一直輪流確認，輪流做，直到全部 process 跑完。
      V. 以上方式中都有可能出現某些時候沒 process 跑的狀況，或要好好判斷該次 process 要跑多久的狀況，故此時我運用了我通過 HW2

online judge 的 scheduler 經驗，來做了一些處理，快速找出下一個可 run 的時間點。

D. 輔助：read, write
   I. 在 RR 時，由於我怕 scheduler 和 child 的時脈會不同，故新增了此溝通機制。若無溝通機制，可能 schedule 跑了 500 個 time unit 後，child 才跑 505 個 time unit 而已，此時 scheduler 才去指定改變 priority，累積下來會使 child 需要較少次的 RR 輪到機會，就能真正跑完，雖然我都把正在跑的 process 的優先度設到最高(SCHED_FIFO)了，不過我是在虛擬機上跑，故我無法確定外面真正的電腦的 CPU 是否真的完全給虛擬機用了，故新增此溝通機制可以保險一些。
   II. 但就算拿掉此溝通機制(大概改動個 10 行左右即可)，整體結果仍會是正確的，我皆有進行測試過，只是沒有溝通機制時，若兩個 process 在理論執行完 time 非常接近時，可能會出現誰先真的執行完不太一定，理由同第一點所說。

E. 我的整體架構是有參考網路上學長的(老師有於 4/29 的線上討論說參考架構是 OK 的)，但我整個 code 是有重新打過的，只有架構一樣，內容都不一樣，且我運用了我 HW2 online judge 的排程經驗(似乎到目前還是只有我通過 OAO)，來減少 scheduler 的運算，使兩邊若不能溝通時，時脈可以盡量一致，加強 system call 的實用性，以及賦予 priority 時，給予要執行的 process 最高優先度(SCHED_FIFO：real time 的優先度)。

2. 版本：
- Using Linux kernel version 4.14.25
- Oracle VM VirtualBox version 6.1
- gcc version 5.4.0

3. 比較實際結果與理論結果

4. 解釋造成差異的原因