B06902038 謝易霖

設計:

先將所有的資料依照 ready time 排序，然後已經 ready 的開始執行並將 priority 降低，依照不同的 policy 來提高優先執行的 priority(這邊我原先 setscheduler 的參數是用 SCHED_FIFO，但時間會有很大誤差，後來改成 SCHED_OTHER 才比較正常，但我不清楚原因)，另外將排程與執行子程序的核心分開。

排程 policy 的部分:

FIFO:當沒有 process 進行中，從 ready 的 process 中選最早的。

PSJF:當有新的 process ready 或是沒有進行中的 process，從 ready 的 process 中選剩餘 exe time 最短的。

RR:用一個 queue 管理，沒有 process 進行中或是前一個 process 執行完一個 time quantum，從 Dequeue 一個 process。經過一個 time quantum 還沒結束的 process 和剛 ready 的 process 則加入 queue。

SJF: 當沒有 process 進行中，從 ready 的 process 中選剩餘 exe time 最短的。當一個 process 結束後，wait 並列出該 process 資訊。

核心版本: Linux 4.14.25

比較實際結果與理論結果:

1. 實際上的執行時間比理論的還要多，而先執行的會花上比預期多更多的時間，原因應該是就算降低了 priority，其他的 process 仍會占用一些 CPU，導致後面 waiting 的 process 也有先稍微執行一些。
2. 比較 RR_1 和 FIFO_1，應該要得出差不多的結果，但實際仍有一段落差，推測是 RR 在 schedule 檢查 switch 的部分花了比較多時間，總時間也有點差距(比起另一組相同測資 PSJF_2 和 SJF_4)。