–1. 設計
Start time 的計時為行程的產生時間
由 Scheduler 和 process 組成 各自享有一個 CPU

Scheduler 不斷檢查 running child 是否終止，且檢查有無 process 的 ready_time 到並生成他們，上面兩種情形 scheduler 都會呼叫 sys_time_now()紀錄間以在該 child 終止時呼叫 sys_print_process。
在 create child 時 execl 前先降低 priority 避免 process 一生成馬上取得 CPU。
決定下一個取得 CPU 的方式依照 scheduling policy 實作。
FIFO SJF PSJF 都可以根據當下是否有 process 在執行，照各自的標準選出最小 ready_t/exec_t 的當作下一個執行 process
RR 紀錄正在執行的 process 外，紀錄上一個 quantum_start_time，依據是否超出時間決定是否換下一個 process 取得 CPU。

–2. 核心版本 Linux 4.14.25

–3. 比較實際結果與理論結果，並解釋造成差異的原因
兩顆 CPU 的時間不一定相等，所以兩邊的時間會有落差，在某些邊界情形就會碰到差異(例如 scheduler CPU 比較快，在 RR 導致 exex_t 剛好 500 的 process 在一個 quantum 跑不完), 不過雖然 start 和 finish 和理論會有明顯的差別，但實際占用 CPU 的 time span 是差不多的(例如 finish 要多等一輪， 但其實多使用 CPU 時間只有一瞬間)，且此情形其實只是讓 marginal case 平移(在理論中的 time quantum 恰小於 exec_t 的 case)。
此差異在 FIFO 與 SJF 幾乎無影響。