Report

1. 設計

主要分成 3 個部分實作，main、scheduler、process。實作內容如下：

main：輸入資料，將資料作適當的轉換後交給 scheduler 處理。

結構 Process(定義在 process.h)紀錄 process 的各變數，n_proc 儲存 process 數量，所有 process 都放在 proc_set 中。

scheduler：定義排程函數，模擬排程過程。

1) 初始化所有 process 並按照 ready time 排序，調高 scheduler 優先度，並交由 CPU 0 執行。

2) 變數 ut_clk 紀錄 scheduler 執行後所經過的 unit time，curr 紀錄執行中的 process 在 proc_set 的 index，n_proc_fin 紀錄已執行完的 process 數。

3) 排程方法實作

FIFO：proc_set 已由 ready time 排序，從 index 最小開始執行，執行完成換下一個 process，直到全部執行結束。

RR：額外宣告 Queue 型態(定義在 sche_que.h)的 rr_que，紀錄目前等待執行的 process。當有 process 在等待且沒有 process 正在執行時，執行在佇列最前端的 process；紀錄 process 執行時間，一旦到 time quantum 即暫停執行，移到佇列最末端。

SJF：在 CPU 空閒時，找到當下可以執行的 process 中，執行時間最短者執行，直到全部執行完成。

PSJF：任何時刻都找可以執行的 process 中，執行時間最短者執行。實作簡化成只有新的 process 加入時，才檢查 process 中執行時間最短者。

process：將準備執行的 process fork 出來，子程序模擬 process 執行，紀錄 process 產生時間和結束時間並輸出，另外定義函式處理分配 CPU、運用 sched_setpriority() 與 set_priority() 提升或降低 process 優先度。

2. 核心

Linux-4.14.25

3. 比較與解釋

執行時間不如預期：

TIME_MEASUREMENT 執行後，計算出來的執行時間不一致，因為所有的 process 都是 500 unit time，理論上應該要接近平均值。起初我的程式在調整優先度時，只使用 sched_setpriority()，私下測試與計算執行時間，在換算回 unit time 後，10 次結果在理論上都要與 500 unit time 相差不多，實際

執行卻落差極大，範圍大約是(350 ~ 720)。我認為可能會與系統執行時資源的競爭有關，嘗試再行調整，原來的 sched_setpriority()以外，另外調整此程式 process 的 nice 值，將 scheduler 與執行中的 process 的 nice 調至-10，其餘 process 調到 19。測試後換算結果每個程式的執行時間都比較接近 500 unit time 大約在(480~600)之間。結論是，在機器上因為會有許多程式同時在競爭系統資源，導致程式並不能以穩定的速率執行，若提高優先度可以減少這種情況發生。修改過後，程式執行結果個人認為比較接近理論結果，但仍然有一定誤差，可能與當下機器的狀態有關，行程數較多可能會導致表現較差。另外由於環境是虛擬機，可能也會因為虛擬機的性能而受到影響。