OS Project-1 Report

設計

前處理

將processes 資料讀取進來後，先分別按照 ready time 或 execution time 做排序，並把排序完的ID 存起來。

Scheduler

在 linux kernel裡面，本身就有內建 FIFO scheduler 跟 RR sheduler, 搭配 SCHED_IDLE 實作 SJF 跟 PSJF。

FIFO:

0. `sched_setscheduler` 的設定： 將其參數中的 `sched_priority` 設為 99。
1. 依照以ready time 做排序的順序，依序fork
2. 在前一個 process 結束之前，其餘的先等待
3. 當process terminate後，讓 ready-time排序下一位的process 可以fork
4. 當最後一個child process也 terminate之後，就結束。

SJF/PSJF:

1. 一樣使用依 ready time 做排序的順序。
2. 當有process 被 fork 時，會先用 queue 暫存起來，等待被執行
3. 當有 process 結束時，會從 queue 中挑 執行時間最短的拿出來執行。
4. 當最後一個child process也 terminate之後，就結束。
5. SJF 和 PSJF 的差別：對 PSJF 來說，會需要再去確認行程剩餘時間，以評估接續要用 SCHED_FIFO 或是 SCHED_IDLE。

system call

`sys_my_time` 是一個用來計時的 system call，當參數 `isStart` 設為1時，代表開始，將會記錄開始時間，反之當 `isStart` 是 0 時，代表結束了，便會記錄結束時間，並將開始與結束 timestamp 輸出。結果可以用 `dmesg | grep project1` 看。

核心版本

Linux 4.14.25

實際與理論比較

以 SJF_1 為例：

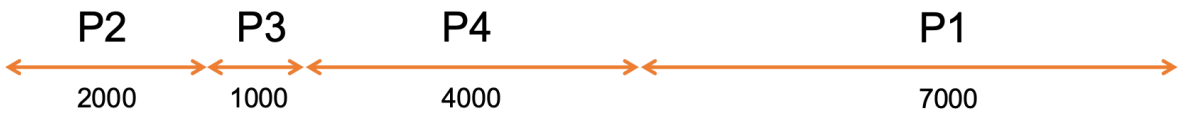
P1 0 7000

P2 0 2000

P3 100 1000

P4 200 4000

理論上其 執行順序與時程如下：



假設將時間基準點放在P2結束時，往後兩兩 process finish time 的時間差應為 1:4:7。

（P3 finish time 距離 P2 finish time → 1000,

P4 finish time 距離 P3 finish time → 4000.. 以此類推）

```
Process: P2 8856
Process: P1 8857
Process: P3 8858
Process: P4 8859
```

```
[236654.058542] [project1] 8856 1588154875.497081964 1588154879.551920357
[236662.540960] [project1] 8858 1588154879.870356912 1588154888.034339172
[236676.289320] [project1] 8859 1588154880.510863022 1588154901.782698777
[236691.691369] [project1] 8857 1588154876.458016207 1588154917.184747244
```

可以看見其process排序的結果是正確的，

然而 dmesg 得到的 finish time ，用和上述相同的方法找出時間間隔：

8482418815 ： 13748359605 ： 15402048467 (nsec)

大約等於 1： 1.6 ：1.8，是與理論不符合的。