B06902109 2020_OS_PJ1

1. 設計

主程序在 CPU0 上執行，其他子程序都在 CPU1 上執行。
先以 sched_setshceduler 把 policy 設成 SCHED_FIFO。

用一個 Linked-List 以表示要執行的工作。List 所用的資料結構為

```
struct cmdlist
{
    UINT time; //表示此工作的時間
    int starttask; //在此時間 fork 一個 ID 為此的 process，無則-1
    int swapin; //將此 ID 的 process 換入 CPU，無則-1
    int swapout; //將此 ID 的 process 換出 CPU，無則-1
};
```

接著開始計算 List:

FIFO:

先將所有 input 以 execution time 由小排至大，然後從 0 時間開始，依序將 start time 符合的 input 放入一個 Queue 裡，並增加 cmdlist 項，其 starttask 值為該 input 在陣列中的索引。假如沒工作的話就 pop queue 然後增加一個 cmdlist 項，其 swapin 值為該 input 在陣列中的索引。

RR:

與前項相同，只是再加上每 500 時間就強制換工作，並將原工作推入 Queue。

SJF:

先將所有 input 以 execution time 由小排至大，然後從 0 時間開始，依序將 start time 符合的 input 放入一個 MinHeap 裡，並增加 cmdlist 項，其 starttask 值為該 input 在陣列中的索引。假如沒工作的話就 pop heap 然後增加一個 cmdlist 項，其 swapin 值為該 input 在陣列中的索引。

PSJF:

與前項相同，只是再加上當 Heap 的 Top 工作的 execution time 小於目前工作的 execution time 時就強制換工作，並將原工作推入 Heap。

最後依照 List 執行 fork 與 sched_setparam 調整 priority 值即可。
（當 sched_setshceduler 的 policy 為 SCHED_FIFO 與 SCHED_RR 時，setpriority 與 nice 無作用，應使用 sched_setparam 或 sched_setshceduler 調整 priority，因為這兩個 policy 為 Real Time Policy，所以 priority 值範圍不同，

且優先權永遠大於使用其他 policy 之 process）

2. Kernel Version

   Linux-5.6.7

   於 linux kernel 5 以上，getnstimeofday 被移除，新的指令為 ktime_get_ts64, 參數為 struct timespec64 而非 struct timespec（在 64 位元系統兩者相同，syscall 可直接傳入 struct timespec*）

3. 差異

   差異的來源來自：

   一、兩個 CPU 的時脈可能不同，導致計時上出現差異。

   二、Fork, sched_setaffinity 與 sched_setparam 勢必為不同的指令，但只要這些指令不是同時執行，Child Process 就不是在創造的當下就換入 CPU1 執行，也不可能是當下就得到應有的 priority 值。