### Part1:

```
rendybig@rendybig-VirtualBox:~$ cd Desktop/
rendybig@rendybig-VirtualBox:~/Desktop$ gcc sched_test.c -lpthr
ead -std=c99 -o sched test
rendybig@rendybig-VirtualBox:~/Desktop$ sudo ./sched_test
[sudo] password for rendybig:
Thread 1 was created
Thread 2 was created
Thread 2 is running
Thread 1 is running
Thread 1 is running
Thread 2 is running
Thread 1 is running
Thread 2 is running
rendybig@rendybig-VirtualBox:~/Desktop$ sudo ./sched_test SCHED
FIF0
Thread 1 was created
Thread 2 was created
Thread 1 is running
Thread 1 is running
Thread 1 is running
Thread 2 is running
Thread 2 is running
Thread 2 is running
```

### Details:

(FIFO)直接將先建立的 thread 的 priority 設為最大(99)即可。 (FIFO)以 sudo 執行很重要,否則無法使用 real-time scheduling。

# Part2:

1. enqueue\_task\_weighted\_rr()

函數將會給予 task 和 rq,將 task 根據 rq 的資料結構重新將 task 放入。 運行 rq->weighted\_rr.nr\_running++ 增加 rq 中的計數,隨後回報該 scheduler 中有沒有存在 task 需要運行。

2. dequeue\_task\_weighted\_rr()

當 task 完成任務時,呼叫 update\_curr\_weighted\_rr 進行統計運行多久,並且更新與 task 相關的排程權重。接著將 task 從 rq 串列中移除,並且更新 scheduler 的 task 計數 (rq->weighted\_rr.nr\_running--;)。

yield\_task\_weighted\_rr()
 直接運行 requeue

4. pick\_next\_task\_weighted\_rr()

當最上層分配一段時間給 scheduler 運行,運行時會調用這個函數,並拿取要執行的 task,但並不用將其移除串列,並執行 next->se.exec\_start = rq-

>clock; 紀錄 task 的開始運行時間,再呼叫 void pdate\_curr\_weighted\_rr(struct rq\*)。若不更新時間,則計算的相對運行時間會錯誤。

5. task\_tick\_weighted\_rr()

當 scheduler 運行時,每隔一段固定的時間會呼叫此函數。若程序執行量超過 scheduler 的設定,則需要更新串列,。

需根據不同的 task 要補充不同的量 p->weighted\_time\_slice,不只要讓這 支程式重進進入串列,同時需要呼叫 set\_tsk\_need\_resched(p),藉以重新呼叫 pick\_next\_task\_weighted\_rr(struct\_rq\*),否則這支程序會運行到結束。

## Result:

### 

Bonus: Shortest Job First Scheduler

- enqueue\_task\_weighted\_rr()
   與 Weighted round-robin 類似。需要 task 時,直接將串列的首元素移除。
- 2. dequeue\_task\_weighted\_rr() 類同 Weighted round-robin。
- 3. requeue\_task\_weighted\_rr()
  在最短工作優先排程中,由於 task 優先權會變動,不方便確定執行過
  一陣子的 task 要移動到哪裡,最簡單的實作方式採用 dequeue 後再一
  次 enqueue。
- 4. yield\_task\_weighted\_rr() 直接 requeue。
- pick\_next\_task\_weighted\_rr()
   類同 Weighted round-robin。
- 6. task\_tick\_weighted\_rr()

在最短工作優先排程中若按照 Weighted round-robin 的寫法且不補充 quantum,則會變成 non-preemptive SJF。相反地,若要實作 preemptive SJF,需要在 check\_preempt\_curr\_weighted\_rr(struct rq\*, struct task\_struct\*, int) 檢查執行的程序是不是串列的首元素,意即是不是最高優先權,如果不是最高優先權,則呼叫 resched\_task() 即可完成。