Homework #2

Team5 B04611015 陳佳佑 、 B04902009 蕭千惠

4th May, 2017

Part I

(1) Set CPU affinity

```
cpu_set_t cpus;
CPU_ZERO(&cpus); // 初始化
CPU_SET(0, &cpus); // 指定一個CPU
sched_setaffinity(0, sizeof(cpu_set_t), &cpus); // 設定由指定的CPU執行
```

(2) Set scheduler

```
struct sched_param main_param;
main_param.sched_priority = sched_get_priority_max(SCHED_FIFO);
sched_setscheduler(getpid(), SCHED_FIFO, &main_param); // 把 scheduling policy 設成 FIFO
```

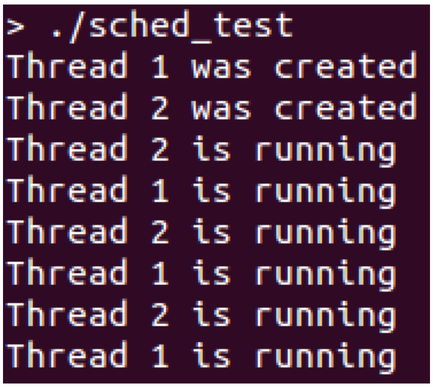
(3) Set the priority of real-time process

```
struct sched_param param[2];
for(int i = 0 ; i < 2 ; i++) {
    // 讓 thread1 所執行的 process 有較高的 scheduling priority
    param[i].sched_priority = sched_get_priority_max(SCHED_FIFO)-i-1;

    // 設定 threads 的 scheduling policy 成 FIFO
    pthread_attr_setinheritsched(&attr[i], PTHREAD_EXPLICIT_SCHED);
    pthread_attr_setschedpolicy(&attr[i], SCHED_FIFO);
    pthread_attr_setschedparam(&attr[i], &param[i]);

    // Create threads
    pthread_create(&threads[i],&attr[i],thread_func,(void *) i);
}
```

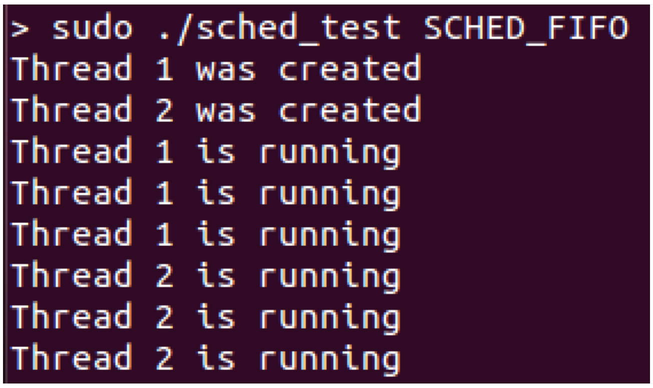
(4) The permission to run real-time process

輸入 `sudo` 以使用 root 的權限來執行 FIFO scheduling.

Result



```
> ./sched_test
Thread 1 was created
Thread 2 was created
Thread 2 is running
Thread 1 is running
Thread 2 is running
Thread 1 is running
Thread 2 is running
Thread 1 is running
```



```
> sudo ./sched_test SCHED_FIFO
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
```

Part II

(1) enqueue_task

```
static void enqueue_task_weighted_rr(struct rq *rq, struct task_struct *p, int wakeup, bool b)
{
    // Add a job into the run queue
     list_add_tail (&(p->weighted_rr_list_item), &(rq->weighted_rr.queue));
    // Renew the number of tasks in the run queue
    rq->weighted_rr.nr_running++;
}
```

1. 把 task 移進 run_queue 中。
2. 把 scheduler 的 task 計數加一，隨後用來在回報該 scheduler 中有沒有存在 task 需要運行。

(2) dequeue_task

```
static void dequeue_task_weighted_rr(struct rq *rq, struct task_struct *p, int sleep)
{
    // First update the task's runtime statistics
    update_curr_weighted_rr(rq);
    // Delete the job from the run queue
     list_del (&(p->weighted_rr_list_item));
    // Renew the number of tasks in the run queue
    rq->weighted_rr.nr_running--;
}
```

1. 完成一項 task 時要呼叫 `update_curr_weighted_rr` 統計執行時間。
2. 把該 task 從 run_queue 中移除。
3. 把 scheduler 的 task 計數減一。

(3) sched_yield

```
static void yield_task_weighted_rr(struct rq *rq)
{
    // Move current task to the tail of the run queue
     list_move_tail (&(rq->curr->weighted_rr_list_item), &(rq->weighted_rr.queue));
}
```

把 current task (`rq->curr`) 放到 queue 的尾端。

(4) pick_next_task

```
static struct task_struct *pick_next_task_weighted_rr(struct rq *rq)
{
    struct task_struct *next;
    struct list_head *queue;
    struct weighted_rr_rq *weighted_rr_rq;

    queue = &((rq->weighted_rr).queue);
    weighted_rr_rq = &(rq->weighted_rr);
    // If there is no task in the run queue, return NULL
    if(rq->weighted_rr.nr_running == 0) return NULL;
    // Pick the hightest priority task in the run queue, which is the first element of the queue
    next = list_first_entry (queue, struct task_struct, weighted_rr_list_item );
    // Record the starting time of the selected task
    next->se.exec_start = rq->clock;
    return next;
}
```

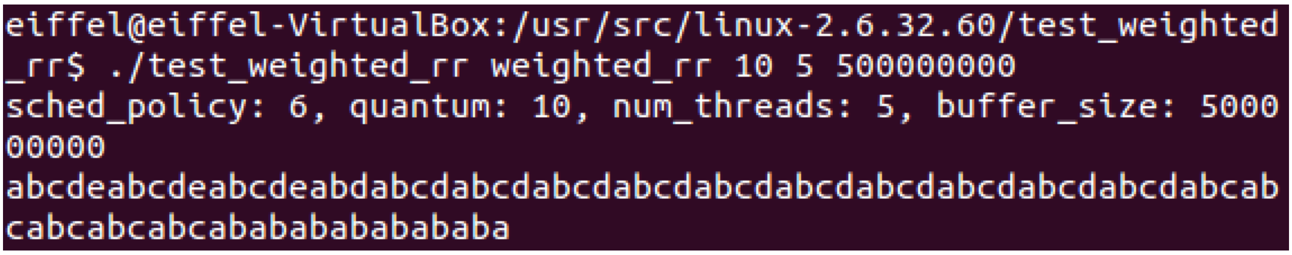
1. 若 run queue 中沒有 task，則回傳 NULL。
2. 挑出下一個要執行的 task。
3. 紀錄 task 的開始運行時間。

(5) task_tick

```
static void task_tick_weighted_rr(struct rq *rq, struct task_struct *p,int queued)
{
    // Update the task's runtime statistics
    update_curr_weighted_rr(rq);
    // Minus one from task_time_slice
    if(p->task_time_slice){
        (p->task_time_slice)--;
        return;
    }

    //.. If task_time_slice is zero
    // First reset task_time_slice to weighted_time_slice
    p->task_time_slice = p->weighted_time_slice;
    // Call set_tsk_need_resched in order to pick next task
    set_tsk_need_resched(p);
    // Yield to next task
    yield_task_weighted_rr(rq);
    return;
}
```

1. 呼叫 `update_curr_weighted_rr` 更新執行時間。
2. 把 `task_time_slice` 減一。
3. 當 `task_time_slice` 變為零時，重設 task 的 `task_time_slice`，並透過呼叫 `set_tsk_need_resched` 藉以重新呼叫 `pick_next_task_weighted_rr`，最後 yield 給下一個 task 執行。

Result



```
eiffel@eiffel-VirtualBox:/usr/src/linux-2.6.32.60/test_weighted
_rr$ ./test_weighted_rr weighted_rr 10 5 500000000
sched_policy: 6, quantum: 10, num_threads: 5, buffer_size: 5000
00000
abcdeabcdeabcdeabdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcab
cabcabcabcabababababababa
```