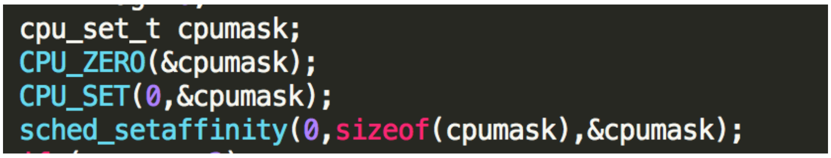
Operating System: Project 2

資工二 B04902051 林承豫
資工二 B04902053 鄭淵仁

Part I

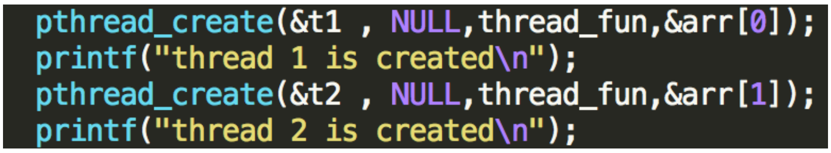
- **Implementation Details**

Step 1. Set the number of CPU to one.



```
cpu_set_t cpumask;
CPU_ZERO(&cpumask);
CPU_SET(0,&cpumask);
sched_setaffinity(0,sizeof(cpumask),&cpumask);
```

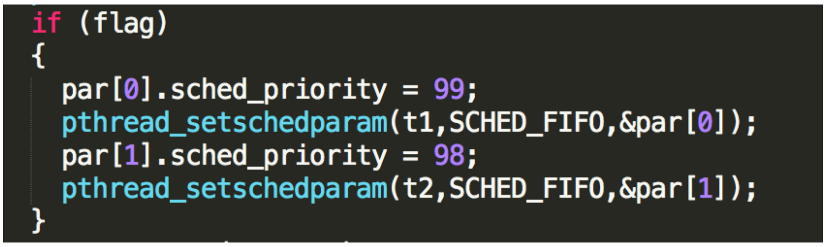
Step 2. Create two threads.



```
pthread_create(&t1 , NULL,thread_fun,&arr[0]);
printf("thread 1 is created\n");
pthread_create(&t2 , NULL,thread_fun,&arr[1]);
printf("thread 2 is created\n");
```

Step 3. If the argument FIFO is input, set the first thread with high priority.

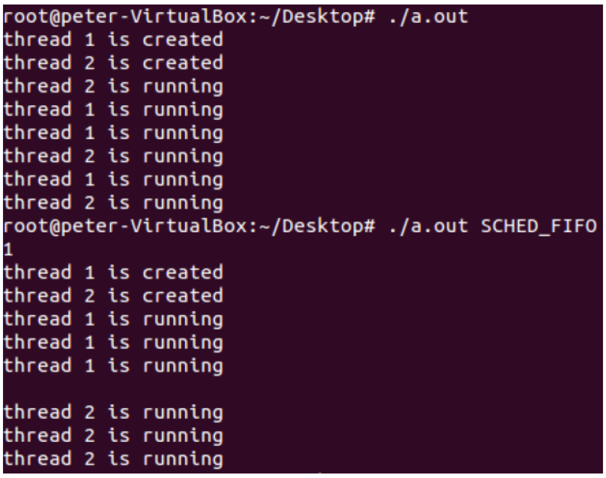
(“flag = 1” means program should be run in FIFO)



```
if (flag)
{
  par[0].sched_priority = 99;
  pthread_setschedparam(t1,SCHED_FIFO,&par[0]);
  par[1].sched_priority = 98;
  pthread_setschedparam(t2,SCHED_FIFO,&par[1]);
}
```

- **Results**

在Linux裡面實際執行結果如下：



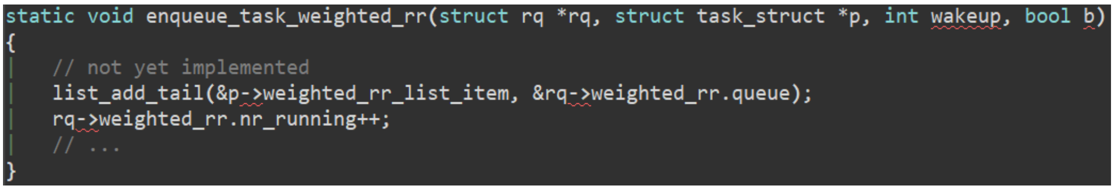
```
root@peter-VirtualBox:~/Desktop# ./a.out
thread 1 is created
thread 2 is created
thread 2 is running
thread 1 is running
thread 1 is running
thread 2 is running
thread 1 is running
thread 2 is running
root@peter-VirtualBox:~/Desktop# ./a.out SCHED_FIFO
1
thread 1 is created
thread 2 is created
thread 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

- **Implementation Details**

1. **enqueue_task_weighted_rr()**

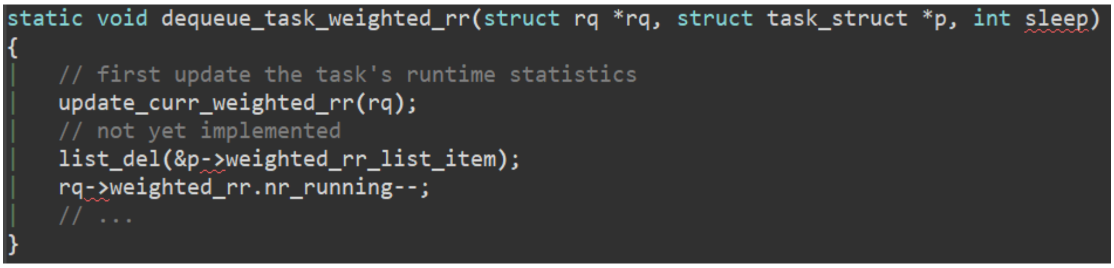
   把task用list_add_tail加到ready queue的最尾端，並把ready queue中的數量加一。程式碼如下圖所示：



```
static void enqueue_task_weighted_rr(struct rq *rq, struct task_struct *p, int wakeup, bool b)
{
    // not yet implemented
    list_add_tail(&p->weighted_rr_list_item, &rq->weighted_rr.queue);
    rq->weighted_rr.nr_running++;
    // ...
}
```

2. **dequeue_task_weighted_rr()**

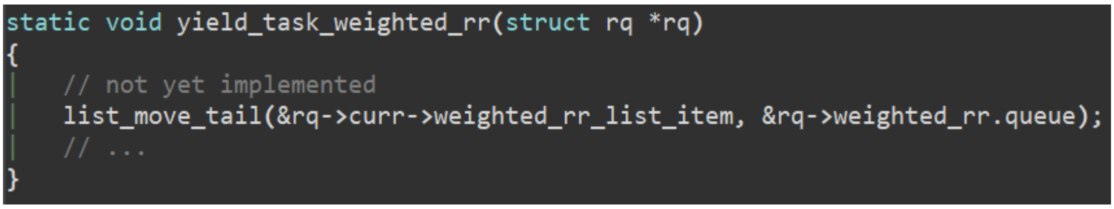
   呼叫update_curr_weighted_rr更新ready queue，並用list_del把task從ready queue中移除，並把ready queue中的數量減一。程式碼如下圖所示：



```
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);
    // not yet implemented
    list_del(&p->weighted_rr_list_item);
    rq->weighted_rr.nr_running--;
    // ...
}
```

3. **yield_task_weighted_rr()**

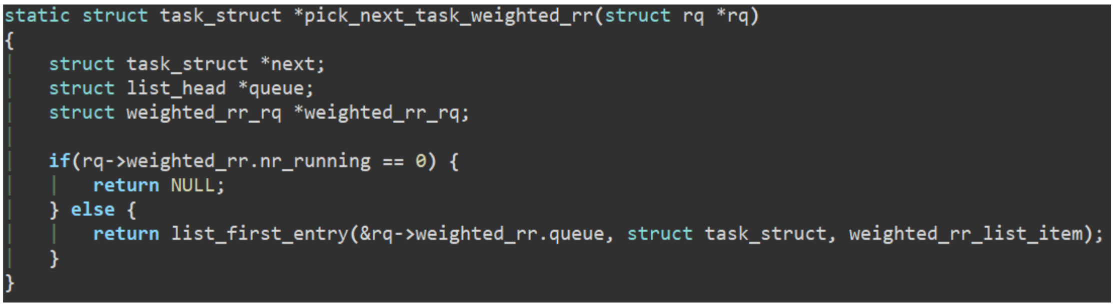
   讓剛跑完的task把cpu資源讓出，且用list_move_tail將剛佔用cpu資源task移到ready queue的最尾端。程式碼如下圖所示：



```
static void yield_task_weighted_rr(struct rq *rq)
{
    // not yet implemented
    list_move_tail(&rq->curr->weighted_rr_list_item, &rq->weighted_rr.queue);
    // ...
}
```

4. **pick_next_task_weighted_rr()**

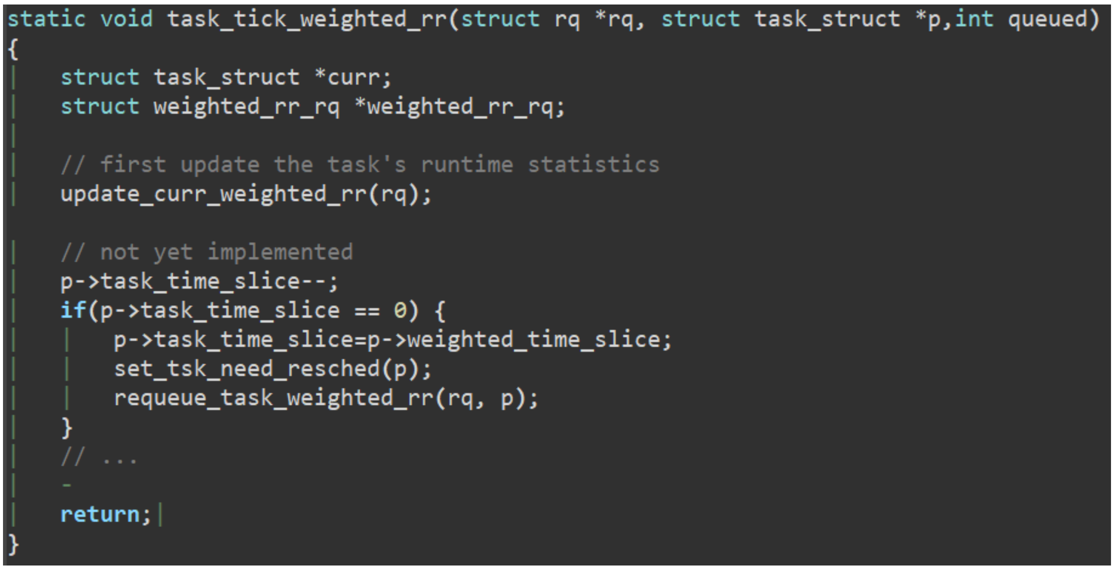
   若 ready queue 中沒有 task 則回傳 NULL，若有則用 list_first_entry 取出 ready queue 中最前面的 task。程式碼如下圖所示：



```
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;

    if(rq->weighted_rr.nr_running == 0) {
        return NULL;
    } else {
        return list_first_entry(&rq->weighted_rr.queue, struct task_struct, weighted_rr_list_item);
    }
}
```

5. task_tick_weighted_rr()

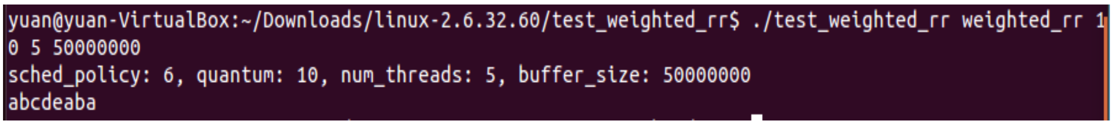
更新現在的 ready queue，將 task 的 time slice 減一，最重要的是若 task 的 time slice 為 0，則更新 time slice，並呼叫 requeue_task_weighted_rr 把 task 排到 ready queue 的最後。程式碼如下圖所示：



```c
static void task_tick_weighted_rr(struct rq *rq, struct task_struct *p,int queued)
{
    struct task_struct *curr;
    struct weighted_rr_rq *weighted_rr_rq;

    // first update the task's runtime statistics
    update_curr_weighted_rr(rq);

    // not yet implemented
    p->task_time_slice--;
    if(p->task_time_slice == 0) {
        p->task_time_slice=p->weighted_time_slice;
        set_tsk_need_resched(p);
        requeue_task_weighted_rr(rq, p);
    }
    // ...

    return;
}
```

- **Result**

在Linux裡面實際執行結果如下：



```
yuan@yuan-VirtualBox:~/Downloads/linux-2.6.32.60/test_weighted_rr$ ./test_weighted_rr weighted_rr 1
0 5 50000000
sched_policy: 6, quantum: 10, num_threads: 5, buffer_size: 50000000
abcdeaba
```