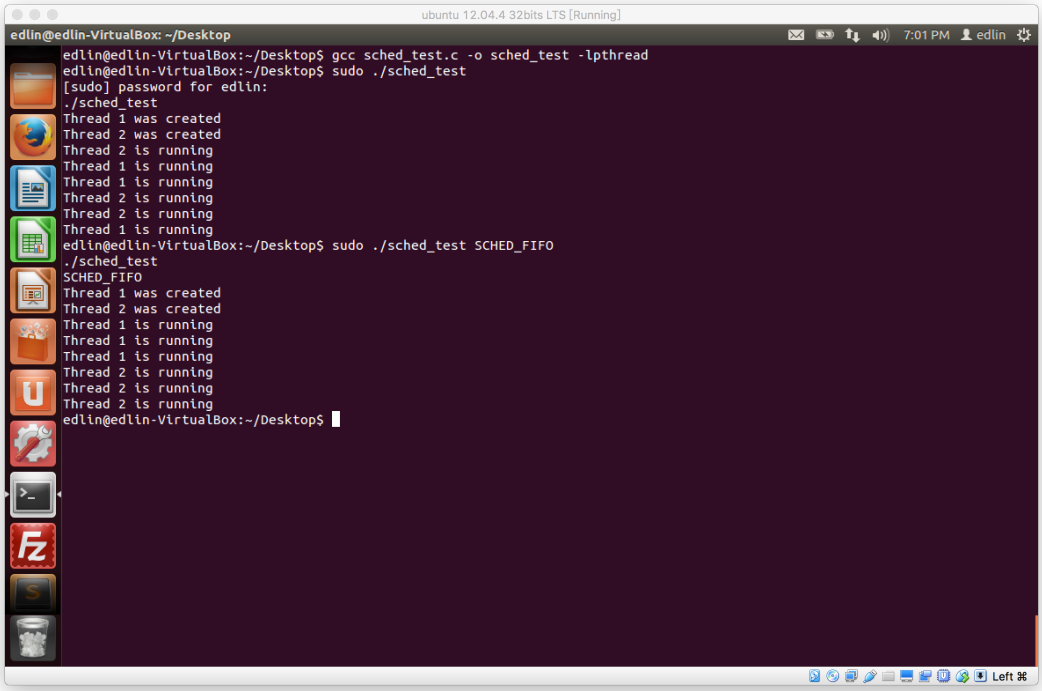
Team 35
資工三 B03902125 林映廷
資工三 B03902127 陳鵬宇

Operation Systems Project2

Part 1 Result



Part 1 Implementation Details

busy waiting :

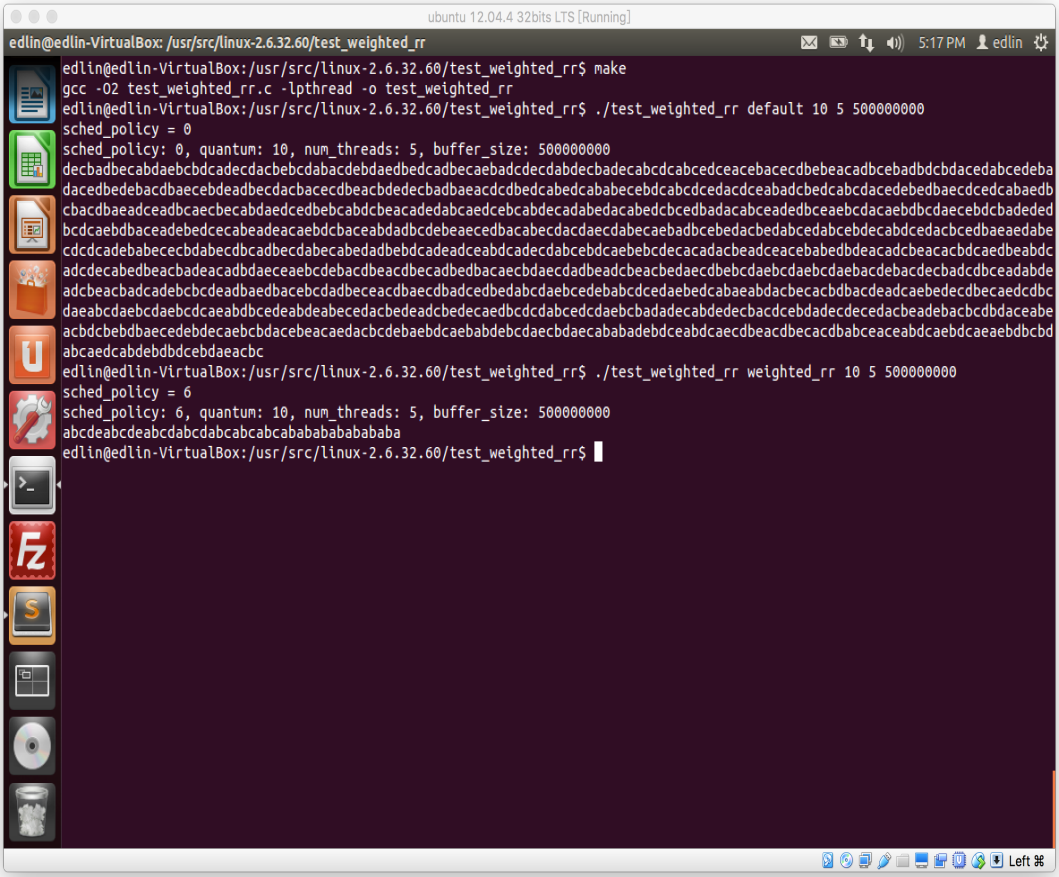
```
void busy(){
    int i;
    for(i = 0; i < 10000000; i++);
    return;
}
```

create & join to threads :

```
for(i = 0; i < 2; i++){
    pthread_create(&tid[i], NULL, thread_func, (void*)&threadno[i]);
    printf("Thread %d was created\n", threadno[i]);
}

for(i = 0; i < 2; i++){
    pthread_join(tid[i], NULL);
}
```

Part 2 Result



Part 2 Implementation Details

enqueue_task_weighted_rr :

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

dequeue_task_weighted_rr :

```
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
    //struct weighted_rr_rq wrr_rq = &(rq->weighted_rr);
    p->task_time_slice = 0;
    list_del(&(p->weighted_rr_list_item));
    rq->weighted_rr.nr_running--;
    // ...
}
```

yield_task_weighted_rr :

```
 */
static void
yield_task_weighted_rr(struct rq *rq)
{
    // not yet implemented
    //struct task_struct *p = rq->curr;
    //p->task_time_slice = p->weighted_time_slice;
    list_move_tail(&(rq->curr->weighted_rr_list_item), &(rq->weighted_rr.queue));
    //requeue_task_weighted_rr(rq, p);
    //set_tsk_need_resched(p);
    // ...
}
```

pick_next_task_weighted_rr :

```
 */
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;


    // not yet implemented
    weighted_rr_rq = &(rq->weighted_rr);
    queue = &(weighted_rr_rq->queue);
    if(list_empty(queue)){
        return NULL;
    }

    next = list_first_entry(queue, struct task_struct, weighted_rr_list_item);
    next->se.exec_start = rq->clock;
    // ...

    /* you need to return the selected task here */
    return next;
}
```

task_tick_weighted_rr :

```
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
    /*if(!task_has_weighted_rr_policy(p)){
        return;
    }*/
    p->task_time_slice--;
    if(p->task_time_slice <= 0){
        p->task_time_slice = p->weighted_time_slice;
        //requeue_task_weighted_rr(rq, p);
        set_tsk_need_resched(p);
        requeue_task_weighted_rr(rq, p);
    }
    // ...

    return;
}
```