OS Project 2

Team02: B04902027 陳昇 , B04902023 鄭士驤

Your implementation details and results

P1:

Details:

呼叫系統自帶的 FIFO 排程

Results:

```
student@student-VirtualBox:~/Desktop/oshw2$ sudo ./a.out
[sudo] password for student:
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
student@student-VirtualBox:~/Desktop/oshw2$ sudo ./a.out 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
student@student-VirtualBox:~/Desktop/oshw2$
```

P2:

Details:

```c
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--;
        // ...
}

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++;
        // ...
}
```

```c
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

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

        // ...

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

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(p->task_time_slice==0){
                 p->task_time_slice=p->weighted_time_slice;
                 set_tsk_need_resched(p);
                 requeue_task_weighted_rr(rq, p);
         }
         p->task_time_slice--;
         // ...

         return;
}

static void
yield_task_weighted_rr(struct rq *rq)
{
         // not yet implemented
         requeue_task_weighted_rr(rq, rq->curr);
         // ...
}
```

Results:

```
student@student-VirtualBox:~/Desktop/oshw2/test_weighted_rr$ ./test_weighted_rr weighted_rr 10 5 50000000
sched_policy: 6, quantum: 10, num_threads: 5, buffer_size: 50000000
//+ set weighted rr scheduling policy
set secheduler...
finish setting secheduler
//+ create the buffer//+ create and start each thread
//+ wait for all threads to complete
//+ print val_buf results
abcdebc
student@student-VirtualBox:~/Desktop/oshw2/test_weighted_rr$
```

```
student@student-VirtualBox:~/Desktop/oshw2/test_weighted_rr$ ./test_weighted_rr weighted_rr 10 5 500000000
sched_policy: 6, quantum: 10, num_threads: 5, buffer_size: 500000000
//+ set weighted rr scheduling policy
set secheduler...
finish setting secheduler
//+ create the buffer//+ create and start each thread
//+ wait for all threads to complete
//+ print val_buf results
abcbabaebdacebdacebdacbdcbacadbcadbcadbcabdcabcabcacbacbabababa
student@student-VirtualBox:~/Desktop/oshw2/test_weighted_rr$
```