OS Project 2 Hints

Advisor: Prof. Tei-Wei Kuo

Speaker: Yu-Chen Lin

Part I

Hints 1/3

- Set CPU affinity

```
thread function:
    for i=1 to 3
        print "Thread # is running"
        busy 0.5 second

main function:

    setp 1: set CPU affinity

    step 2: invoke FIFO scheduler

    for i=1 to 2
        pthread_create(i)
        print "Thread # was created"

    for i=1 to 2
        pthread_join(i)
```

Hints:

- Use the function sched_setaffinity() to specify one core that can be used by the scheduler and the two threads
- There are several parameters should be initialized before the sched_setaffinity() function call

Hints 2/3

```
thread function:
    for i=1 to 3
        print "Thread # is running"
        busy 0.5 second

main function:

    setp 1: set CPU affinity

    step 2: invoke FIFO scheduler

    for i=1 to 2
        pthread_create(i)
        print "Thread # was created"

    for i=1 to 2
        pthread_join(i)
```

- Invoke FIFO scheduler

Hints:

- Use the function sched_setscheduler() to change the scheduling policy
- There are several parameters should be initialized before the sched_setscheduler() function call

Hints 3/3

- Busy waiting

Discussion:

- sleep(500)???

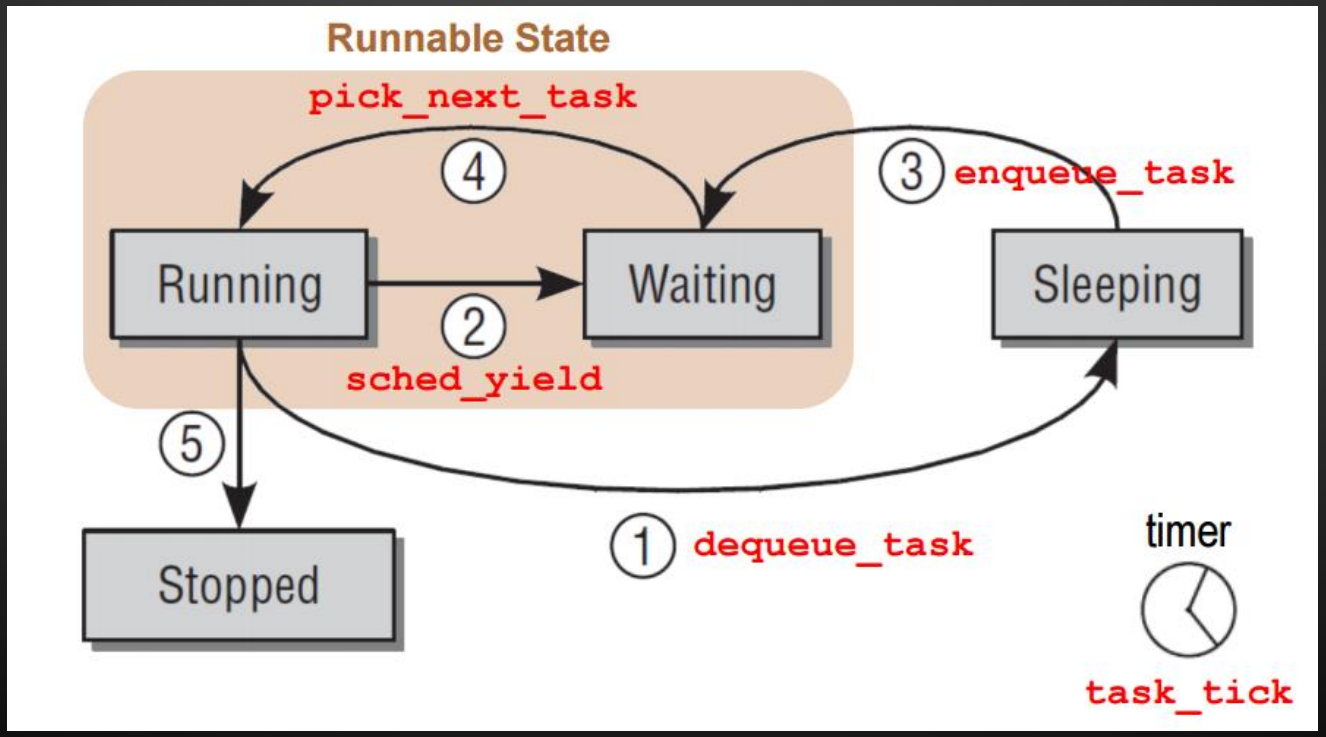
```
thread function:
    for i=1 to 3
        print "Thread # is running"
        busy 0.5 second

main function:

    setp 1: set CPU affinity

    step 2: invoke FIFO scheduler

    for i=1 to 2
        pthread_create(i)
        print "Thread # was created"

    for i=1 to 2
        pthread_join(i)
```

Part II

In linux-2.6.32.60/kernel/sched_weighted_rr.c

- Accomplish the five functions of weighted rr scheduler
  - static void enqueue_task_weighted_rrstatic void (struct rq *rq, struct task_struct *p, int wakeup, bool b)
  - static void dequeue_task_weighted_rr (struct rq *rq, struct task_struct *p, int sleep)
  - static void yield_task_weighted_rr (struct rq *rq)
  - static struct task_struct *pick_next_task_weighted_rr (struct rq *rq)
  - static void task_tick_weighted_rr (struct rq *rq, struct task_struct *p, int queued)

Relationships between Generics Functions and Process States



Hints 1/4

- static void enqueue_task_weighted_rrstatic void (struct rq *rq, struct task_struct *p, int wakeup, bool b)
- static void dequeue_task_weighted_rr (struct rq *rq, struct task_struct *p, int sleep)

Hints:

- Use functions list_add_tail() and list_del() to enqueue and dequeue task_struct *p
- Remember to update the rq->weighted_rr.nr_running value after enqueuing/dequeuing

Hints 2/4

- static void yield_task_weighted_rr (struct rq *rq)

Hint:

- Use the function list_move_tail() to put the current task (rq->curr) to the end of the run list

Hints 3/4

- static void task_tick_weighted_rr
  (struct rq *rq, struct task_struct *p, int queued)

Hints:

- task_tick is called by the periodic scheduler each time it is activated
- First, task_time_slice value of the task p minus one
- Once task_time_slice value of the task p is zero
  1. reset task_time_slice of the task p
  2. call the function set_tsk_need_resched(q)
  3. yield/requeue the task p

Hints 3/4 (cont.)

In linux-2.6.32.60/include/linux/sched.h

- task_time_slice:
  - record the consumption of time slice
- weighted_time_slice:
  - how much time should be supplied when reset task_time_slice is according to the weighted_time_slice

```
1219 struct task_struct {
1220     ...
1221     //+ RTS Proj2: weighted_rr
1222     unsigned int task_time_slice;
1223     unsigned int weighted_time_slice;
1224     ...
1225         //+ RTS Proj2: weighted_rr
1226     struct list_head weighted_rr_list_item;
```

Hints 3/4 (cont.)

In linux-2.6.32.60/kernel/sched.c

```
static void __sched_fork(struct task_struct *p)
{
    ...
    //+ OS  Proj2: weighted_rr
    INIT_LIST_HEAD(&p->weighted_rr_list_item);
    p->task_time_slice = weighted_rr_time_slice;
    p->weighted_time_slice = weighted_rr_time_slice;
    ...
}

//+ OS  Proj2: weighted_rr
SYSCALL_DEFINE1(sched_weighted_rr_setquantum, unsigned int, quantum)
{
    weighted_rr_time_slice = quantum;
    return;
}
```

Hints 4/4

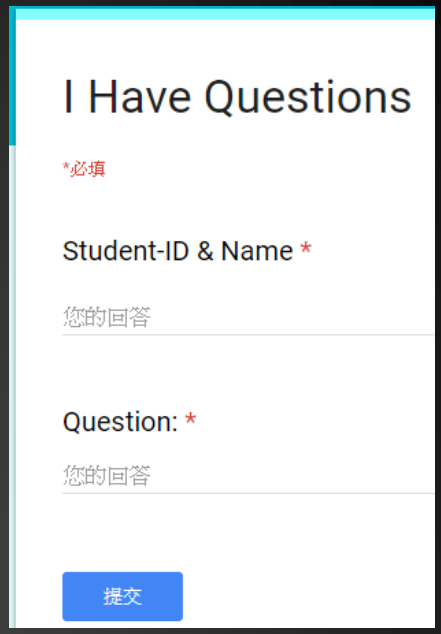
- static struct task_struct *pick_next_task_weighted_rr (struct rq *rq)
- pick_next_task selects the next task that is supposed to run, while put_prev_task is called before the currently executing task is replaced with another one

Hints:

- If weighted_rr.queue is empty, return NULL (determined by weighted_rr.nr_running value)
- Otherwise, use the function list_first_entry() to obtain and return the first entry/task in weighted_rr.queue

Contact TAs

- If you have any question about the project, please feel free to contact TAs.
- I have questions: https://goo.gl/forms/39eB4ex4w3EX7I4K2
- Video: http://newslab.csie.ntu.edu.tw/course/OS2018/PJ2_Hint.html
- Han-Yi Lin: d03922006@csie.ntu.edu.tw
  Yu-Chen Lin: f04922077@csie.ntu.edu.tw
  Yi-Shen Chen: d05922009@csie.ntu.edu.tw
  Yu-Chuan Chang: r05922057@csie.ntu.edu.tw



References

- Reference Book
  - Professional Linux® Kernel Architecture, Wolfgang Mauerer, Wiley Publishing, Inc.

- Process Scheduling
  - https://www.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html