Notes

- The next week class (5/27) will be moved to R104, since R103 is reserved for holding a workshop.
- Reference book has been uploaded to the facebook group, and the decompress password is ``useforstudy''
- Midterm exam paper sheets and project 3 will be released at 5/27 class

OS Project 2
Hints

Advisor: Tei-Wei Kuo
Speaker: Chien-Chung Ho

Scheduler Classes Side (2/3)

As well in “kernel/sched.c”,

- Define simple_rr_rq structure, which should contain
  - struct list_head queue – to denote the actual run queue for your simple rr scheduler
  - unsigned long nr_running – to denote the number of processes which are now in the run queue

```
//+ OS Proj2: simple_rr
/* SIMPLE_RR classes' related field in a runqueue: */
struct simple_rr_rq {
    struct list_head queue;
    unsigned long nr_running;
};
```

Task Side

Generic Scheduler

Scheduler Classes

Task | Task | Task

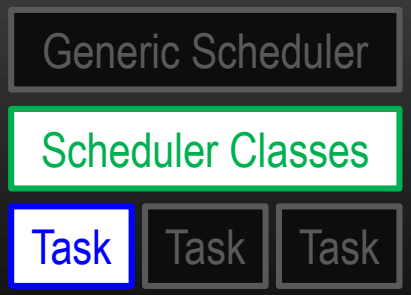
In “struct task_struct” of “include/linux/sched.h”, add

- Declare unsigned int simple_rr_task_time_slice – to denote the current time slice for this task
- Declare struct list_head simple_rr_list_item – to denote the list item which will be inserted into the run queue of simple_rr

```
struct task_struct {
    ...
    //+ OS Proj2: simple_rr
    struct list_head simple_rr_run_list;
    //+ OS Proj2: simple_rr
    unsigned int task_time_slice;
    ...
```

simple_rr_list_item

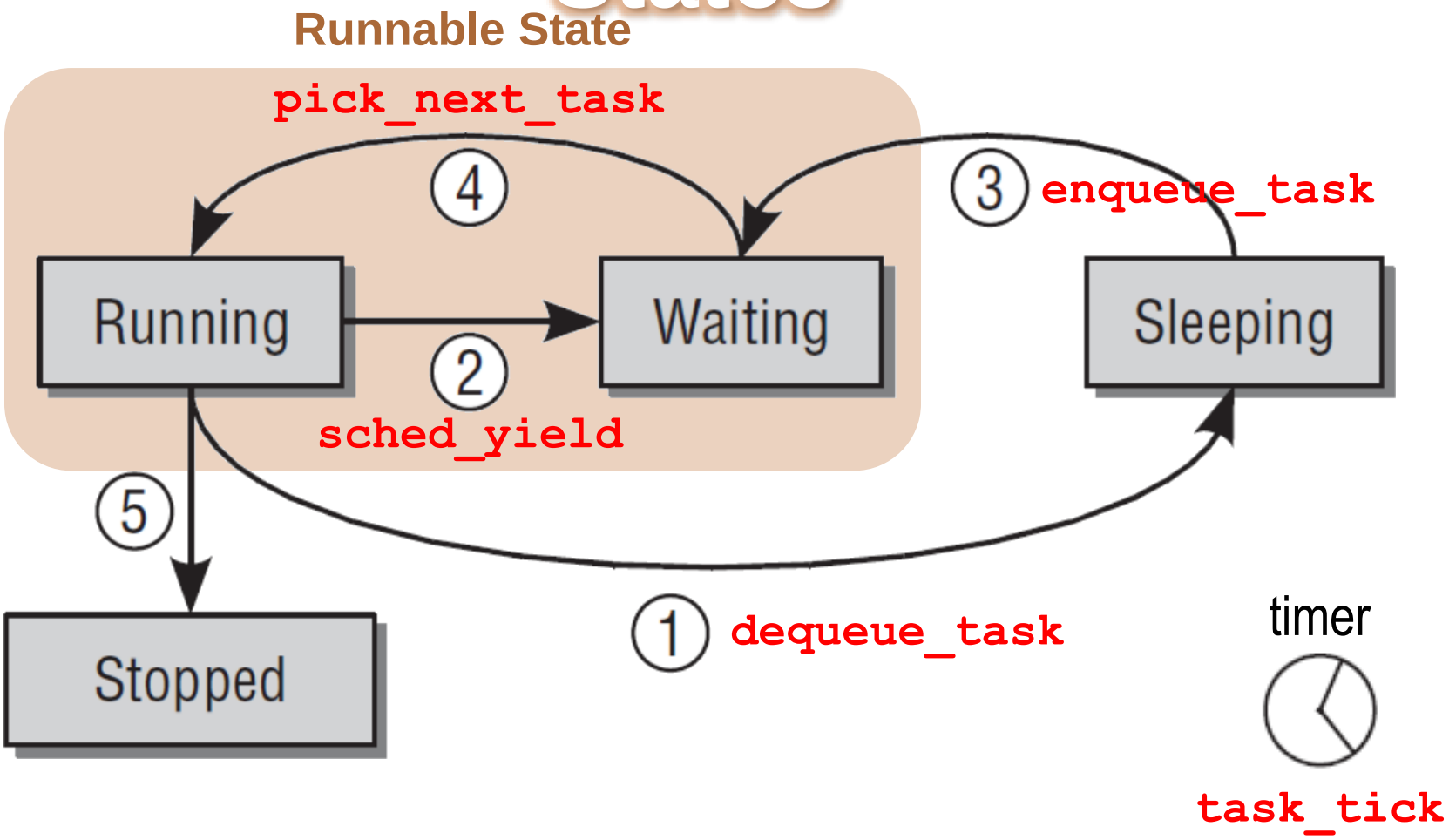
Your Requirements



In “kernel/sched_simple_rr.c”

- Accomplish the five functions of simple rrscheduler
  - static void enqueue_task_simple_rr(structrq*rq, structtask_struct*p, int wakeup, bool b)
  - static void dequeue_task_simple_rr(structrq*rq, structtask_struct*p, int sleep)
  - static void yield_task_simple_rr(struct rq *rq)
  - static struct task_struct*pick_next_task_simple_rr(struct rq *rq)
  - static void task_tick_simple_rr(structrq*rq, struct task_struct*p,int queued)

Relationships between Generics Functions and Process States



Hints (1/4)

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

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

Hints (2/4)

- static void yield_task_simple_rr(structrq*rq)
- Use the function list_move_tail to put the current task to the end of the run list

Hints (3/4)

- static void task_tick_simple_rr(structrq*rq, structtask_struct*p,intqueued)
- task_tick is called by the periodic scheduler each time it is activated.

- 1) 每次對p->task_time_slice的值做減一的動作
- 2) 當p->task_time_slice的值到零
  - 2.1) 重設p->task_time_slice
  - 2.2) set_tsk_need_resched(q)
  - 2.3) yield/requeue該task

Hints (4/4)

- static struct task_struct* pick_next_task_simple_rr(structrq*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.
- 1) 若simple_rr.queue是空的, 回傳NULL
- 2) 否則,
  - 2.1)用list_first_entry()去取得並回傳simple_rr.queue中的第一筆entry/task
  - 2.2) 記得要update 該task 的se.exec_start值(hint: rq->clock)

Scoring of Project 2 (1/2)

- Implement the below FIVE incomplete functions in "simple_rr.c" (50%)
  - enqueue_task_simple_rr()
  - dequeue_task_simple_rr()
  - yield_task_simple_rr()
  - pick_next_task_simple_rr()
  - task_tick_simple_rr()
- Add a system call to let the test program can set different simple_rr_time_slice values (10%)

Scoring of Project 2 (2/2)

- Report (40%)
  - Your implementation details
  - At most 4 pages
- Bonus (at most 20%)
  - Any variation of the round-robin scheduling policy
    - E.g., priority-weighted round-robin, SJF, and so on.

Submission Rules

- Project deadline: 2015/05/27 (Wednesday) 23:59
  - Delayed submissions yield severe point deduction
- Upload your team project to the FTP site.
  - FTP server: 140.112.28.118 (os2015ktw / ktw2015os)
- The team project should
  - Contain the whole “linux2.6.32.60/” directory
  - Contain your modified test program
  - Contain your report (PDF or DOC, within 4 pages)
  - Be packed as one file named “OSPJ2_Group##.tar.xz”
- **DO NOT COPY THE HOMEWORK**

References

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

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