2016025469 컴퓨터공학과 서건식

운영 체제 HW#9

제출 일자 : 2020/05/28

1. 리눅스 스케줄러 전체 동작과 myrr 설명

   리눅스 스케줄러의 전체동작 :

   전체적인 스케줄링 과정은 다음과 같다 Exit, wait 상태의 태스크들이

   RUNNING state 로 전환하게 되면 rq 에 enqueue_task() 함수를 사용해서

   삽입해 준다 Pick_next_task() 함수로 rq 의 태스크 중 다음으로 수행할 태

   스크를 산출하여 rq -->curr 로 해주고 이를 실행한다 Rq -->curr 상태의 태

   스크가 RUNNING 상태로 돌아올 땐 put_prev_task(), EXIT,WAIT 등 다른

   상태로 돌아갈 땐 dequeue_task() 함수를 호출한다

   Myrr 설명:

   해당 스케쥴러는 Round-Robin 방식을 채택했으며, weight를 따로 주지 않고 모든 프로세스에 공통적인 time quantum을 분배한다. Round-Robin 방식은 각 프로세스의 버스트타임이 배정되어 있는데, 시간할당량 만큼 프로세스가 돌도록 스케쥴링 하는것이다. 이렇게 하면 버스트타임-시간할당량이 리스케쥴링 이후 해당 프로세스의 버스트타임이 된다.

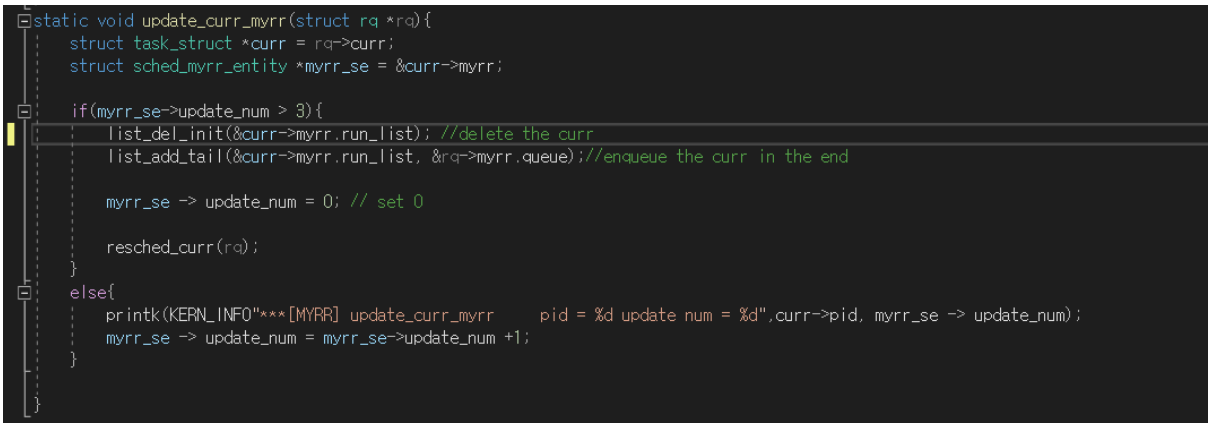
   동작방식은 rq의 list에서, 가장 첫 프로세스가 time quantum만큼 시간을

사용했다면 해당 프로세스를 빼고 맨 뒤에 붙이는 형식이다.

2. myrr.c 설명

myrr.c 설명은 가시성을 위해 윈도우로 옮긴 후 visual studio 화면을 보이도록 하겠습니다.

1) Update_curr_myrr



```
static void update_curr_myrr(struct rq *rq){
    struct task_struct *curr = rq->curr;
    struct sched_myrr_entity *myrr_se = &curr->myrr;

    if(myrr_se->update_num > 3){
        list_del_init(&curr->myrr.run_list); //delete the curr
        list_add_tail(&curr->myrr.run_list, &rq->myrr.queue);//enqueue the curr in the end

        myrr_se -> update_num = 0; // set 0

        resched_curr(rq);
    }
    else{
        printk(KERN_INFO"***[MYRR] update_curr_myrr     pid = %d update num = %d",curr->pid, myrr_se -> update_num);
        myrr_se -> update_num = myrr_se->update_num +1;
    }

}
```

먼저 curr process의 task_struct를 저장합니다.

Sched_myrr_entity 구조체인 myrr_se를 선언하고, curr->myrr을 저장합니다.

조건문에서는 myrr_se의 update_num 상태에 따라 다르게 동작합니다. 먼저 update_num이 3보다 작은 경우 else문으로 들어가 update_num에 +1을 해줍니다. 3보다 큰 경우 if문으로 들어가서 list_del_init 함수를 사용해 현재 프로세스를 지워준 후에, list_add_tail로 지운 프로세스를 리스트의 마지막에 추가해줍니다.

Update_num을 0으로 초기화 해준후에, resched_curr()을 호출하여 다시 스케쥴링합니다.

2) Enqueue_task_myrr()

```
static void enqueue_task_myrr(struct rq *rq, struct task_struct *p, int flags) {

    list_add_tail(&p->myrr.run_list, &rq->myrr.queue);
    rq->myrr.nr_running++;
    printk(KERN_INFO"***[MYRR] enqueue: success cpu=%d, nr_running=%d, pid=%d₩n",cpu_of(rq), rq->myrr.nr_running,p->pid);

}
```

Myrr.queue의 끝에 p를 삽입하고 nr_running의 값을 증가시킵니다.

3) Dequeue_task_myrr()

```
static void dequeue_task_myrr(struct rq *rq, struct task_struct *p, int flags)
{
    if(rq->myrr.nr_running>0)
    {
        list_del_init(&p->myrr.run_list);
        rq->myrr.nr_running--;
        printk(KERN_INFO"₩t***[MYRR] dequeue: success cpu=%d, nr_running=%d, pid=%d₩n",cpu_of(rq), rq->myrr.nr_running,p->pid);
    }
    else{
    }

}
```

nr_running의 값이 0보다 크면 list_del_init함수로 프로세스를 지워주고 nr_running의 값을 감소시킵니다.

4) pick_next_task_myrr()

```
struct task_struct *pick_next_task_myrr(struct rq *rq, struct task_struct *prev)
{
    struct task_struct *p;
    struct sched_myrr_entity *myrr_se = NULL;

    if(rq->myrr.nr_running==0 ){
        return NULL;
    }
    else{
        myrr_se = list_entry(rq->myrr.queue.next, struct sched_myrr_entity,run_list);

        p = container_of(myrr_se, struct task_struct, myrr);


        printk(KERN_INFO "₩t***[MYRR] pick_next_task: cpu=%d, prev->pid=%d,next_p->pid=%d,nr_running=%d₩n",cpu_of(rq),prev->pid,p->pid,rq->myrr.nr_running);
        return p;
    }
}
```

Nr_running이 0이 아닌 경우 else문으로 들어가서 현재 queue의 next를 sched_myrr_entity타입으로 가져온 후에 이를 container_of 함수를 사용해 p에 저장한 후 return 합니다.

3. 최종 결과

Dmesg 결과:

```
[   51.252723]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1880,next_p->pid=1883,nr_running=4
[   51.253764] ***[MYRR] update_curr_myrr               pid = 1883 update num = 0
[   51.255343] ***[MYRR] update_curr_myrr               pid = 1883 update num = 1
[   51.255818] ***[MYRR] update_curr_myrr               pid = 1883 update num = 2
[   51.256707] ***[MYRR] update_curr_myrr               pid = 1883 update num = 3
[   51.257820]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1883,next_p->pid=1881,nr_running=4
[   51.258713] ***[MYRR] update_curr_myrr               pid = 1881 update num = 0
[   51.259684] ***[MYRR] update_curr_myrr               pid = 1881 update num = 1
[   51.260672] ***[MYRR] update_curr_myrr               pid = 1881 update num = 2
[   51.261940] ***[MYRR] update_curr_myrr               pid = 1881 update num = 3
[   51.263648] ***[MYRR] update_curr_myrr               pid = 1881 update num = 0
[   51.263770]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1881,next_p->pid=1882,nr_running=4
[   51.264884] ***[MYRR] update_curr_myrr               pid = 1882 update num = 0
[   51.265886] ***[MYRR] update_curr_myrr               pid = 1882 update num = 1
[   51.267191] ***[MYRR] update_curr_myrr               pid = 1882 update num = 2
[   51.267596] ***[MYRR] update_curr_myrr               pid = 1882 update num = 3
[   51.268979]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1882,next_p->pid=1880,nr_running=4
[   51.269748] ***[MYRR] update_curr_myrr               pid = 1880 update num = 0
[   51.270871] ***[MYRR] update_curr_myrr               pid = 1880 update num = 1
[   51.271314] ***[MYRR] update_curr_myrr               pid = 1880 update num = 2
[   51.272619] ***[MYRR] update_curr_myrr               pid = 1880 update num = 3
[   51.273945]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1880,next_p->pid=1883,nr_running=4
[   51.274769] ***[MYRR] update_curr_myrr               pid = 1883 update num = 0
[   51.275756] ***[MYRR] update_curr_myrr               pid = 1883 update num = 1
[   51.276758] ***[MYRR] update_curr_myrr               pid = 1883 update num = 2
[   51.277744] ***[MYRR] update_curr_myrr               pid = 1883 update num = 3
[   51.278902]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1883,next_p->pid=1881,nr_running=4
[   51.279731] ***[MYRR] update_curr_myrr               pid = 1881 update num = 1
[   51.280711] ***[MYRR] update_curr_myrr               pid = 1881 update num = 2
[   51.281702] ***[MYRR] update_curr_myrr               pid = 1881 update num = 3
[   51.282843]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1881,next_p->pid=1882,nr_running=4
[   51.283689] ***[MYRR] update_curr_myrr               pid = 1882 update num = 0
[   51.284671] ***[MYRR] update_curr_myrr               pid = 1882 update num = 1
[   51.285664] ***[MYRR] update_curr_myrr               pid = 1882 update num = 2
[   51.286840] ***[MYRR] update_curr_myrr               pid = 1882 update num = 3
[   51.288950] ***[MYRR] update_curr_myrr               pid = 1882 update num = 0
[   51.289093]  ***[MYRR] pick_next_task: cpu=1, prev->pid=1882,next_p->pid=1880,nr_running=4
[   51.289841] ***[MYRR] update_curr_myrr               pid = 1880 update num = 0
[   51.290822] ***[MYRR] update_curr_myrr               pid = 1880 update num = 1
```

스케줄러가 update_curr_myrr 함수 호출이 3번 될 때 마다 다시 스케줄링하여 다른 프로세스를 스케줄링 함을 볼 수 있습니다. (update_num은 0으로 초기화하여 시작 숫자가 0이라서, 실습 수업 pdf의 출력결과(2)와 시작숫자가 다릅니다.)

Kill `pidof newclass` 명령어는 약 10초뒤에 입력하였습니다.

ova파일 구글 드라이브 링크:

https://drive.google.com/open?id=1xKH_VMr9bzPkmxxb8bFfKBPDrWChrjn9