**2017029589 컴퓨터소프트웨어학부 류지범**

**운영 체제 HW#9**

**제출 일자 : 2020/05/30**

1. **과제 A**
2. **리눅스 스케줄러 전체 동작과 myrr 설명**

우선 스케줄러의 우선순위는 rt>mysched>myrr>fair>idle이다.

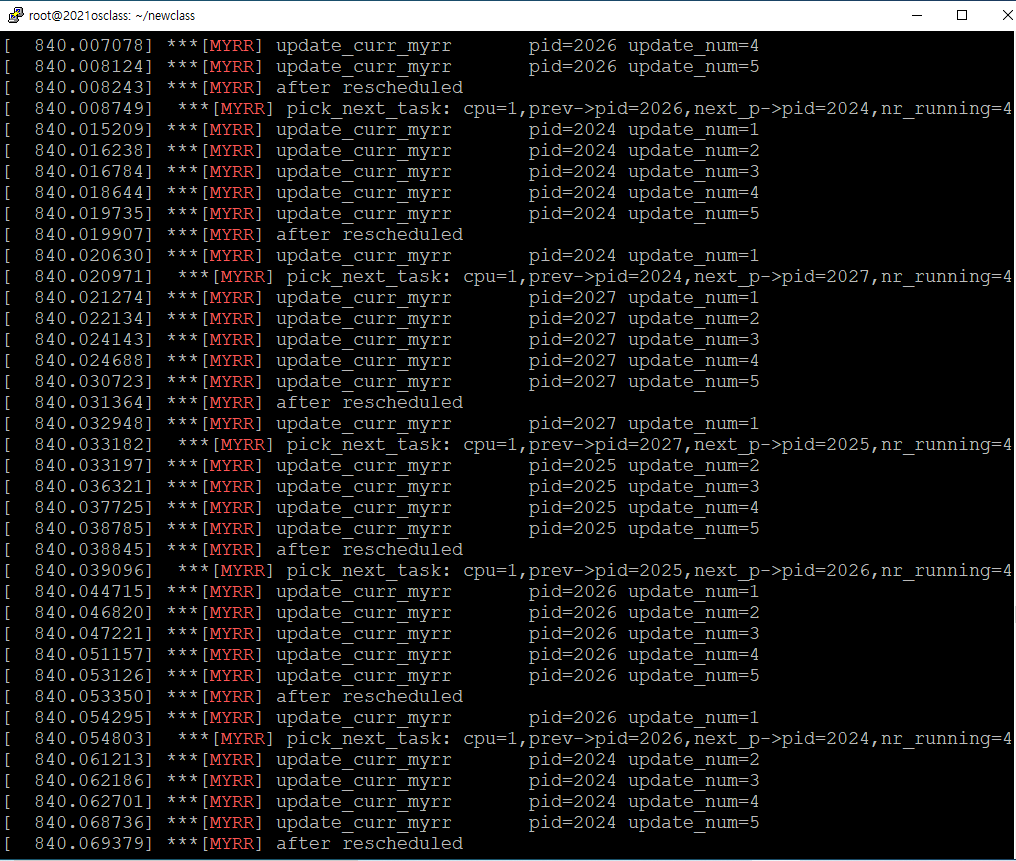
원래 round robin 방식은 time slice만큼 cpu를 할당해주는 스케줄링 알고리즘이고, 프로세스가 time slice를 다 소진하면 맨 뒤로 이동하는 방식을 가지고 있다. 우리가 구현한 myrr은 time slice를 tick으로 계산해서 사용하지 않고 update\_num을 1씩 증가시키면서 사용한다.

Myrr의 경우 newclass2 코드에 의해 1초 간격으로 fork 된다. 이때 update\_num이 4보다 크면 update\_num을 0으로 초기화하고, 현재 실행중인 프로세스를 런큐의 맨 뒤로 옮기고 reschedule 함수를 호출해서 스케줄링을 실행한다.

1. **myrr.c 설명**

우선 myrr의 기본적인 코드는 mysched와 거의 비슷하다. Round robin방식이 FIFO의 방식에 기반하기 때문에 enqueue와 dequeue, pick\_next\_task의 코드는 mysched와 동일하게 작성했다. 프로세스가 들어온 순서대로 큐의 맨 뒤에 추가하고, 먼저 들어온 프로세스 순서대로 실행하도록 하는 방식을 사용했다.

우리가 구현한 MYRR은 time slice(update\_num)를 초과하면 rescheduling이 되도록 하는 것이다. Time slice를 체크하기 위해, task\_tick\_myrr에서 tick이 발생할 때마다 update\_curr\_myrr을 호출하도록 해주었다. Update\_curr\_myrr이 실행되면 우선 현재 sched\_class가 myrr이 아니면 바로 리턴해주도록 했고 아닐 경우, 해당 entity의 update\_num을 1 증가시켜준다. 그리고 현재 프로세스의 pid와 update\_num을 출력해준다. 그 후 update\_num이 time\_slice(4)를 초과할 경우 list\_move\_tail을 사용해서 현재 프로세스를 런큐의 제일 마지막으로 이동시켜준다. 이동시키고 난 후 update\_num을 0으로 초기화하고 resched\_curr함수를 호출해서 리스케줄이 되도록 해준다.

1. **최종 결과**

최종 결과는 다음과 같은 결과를 보여준다. Update\_num이 4를 초과할 경우 리스케줄되고 런큐의 가장 맨 뒤로 이동한다.

결과를 보면 정확하게 round robin방식을 만족시키지 않고 있는데, 12345가 연속적으로 나오는 것이 아니라 끊기는 경우가 발생한다. 이 이유는 우선순위가 더 높은 스케줄러에 의해 preemption되기 때문이다. 선점당하기 때문에 우리가 원하는 결과가 잘 나오지 않을 수 있다.

만약에 숫자를 큰 값을 줬을 경우엔 FIFO와 동일하게 동작하게 될 것이다. 들어온 순서대로 프로세스가 모든 할 일을 다 하고 dequeue되는 것이 반복될 것이므로 FIFO와 동일하게 동작한다.

가상시스템 ova 링크는 다음과 같다.

https://drive.google.com/file/d/1Q\_J\_3LIqOgAuc95klDZHbe2oSgK7-ngZ/view?usp=sharing