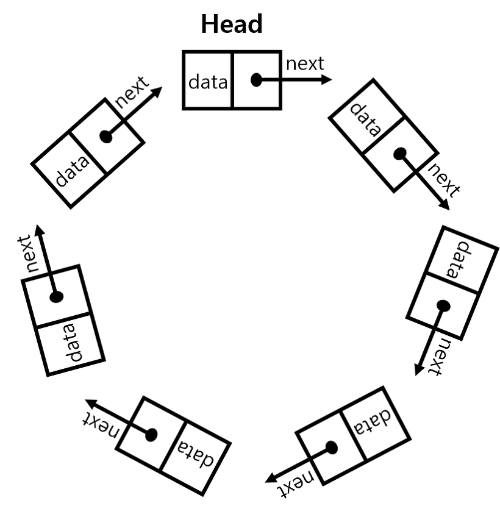
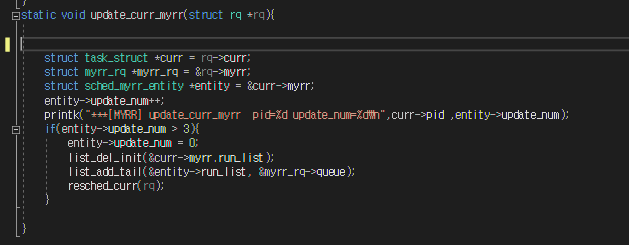
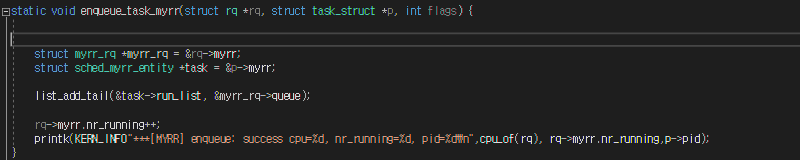
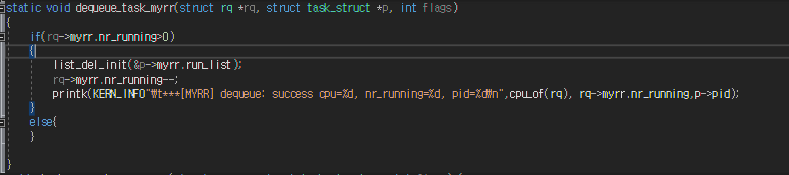
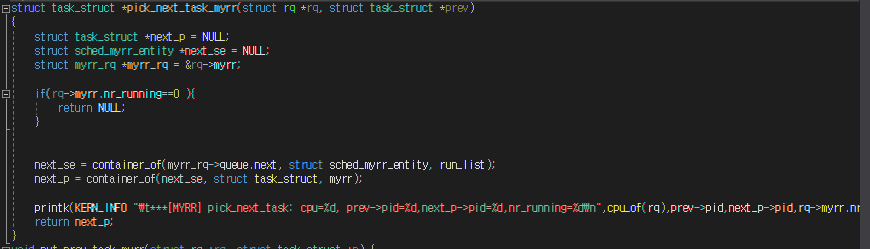
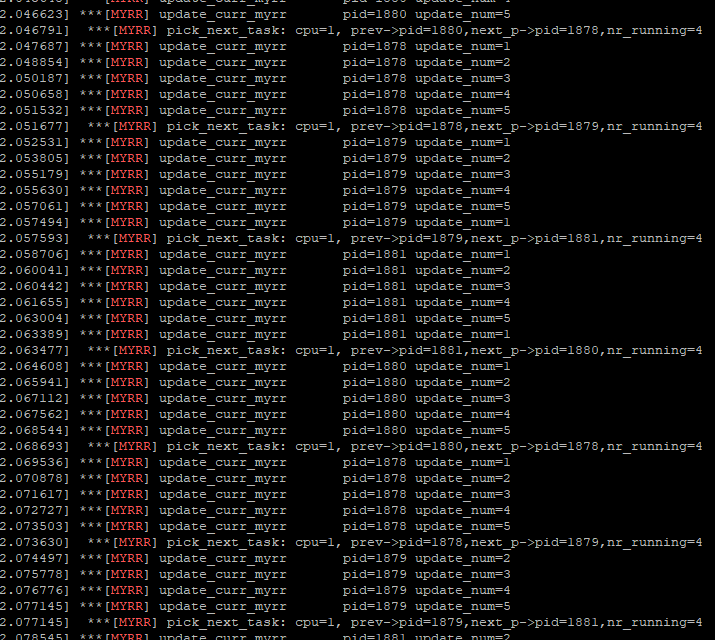
**2019062833 컴퓨터소프트웨어학부 김유진**

**운영 체제 HW#9**

**제출 일자 : 2021-05-27**

1. **과제 A(가상머신 링크:** [**https://drive.google.com/file/d/1bwdcjd3hNQLZ7u9aSRRQyqSJmwXXtqeu/view?usp=sharing**](https://drive.google.com/file/d/1bwdcjd3hNQLZ7u9aSRRQyqSJmwXXtqeu/view?usp=sharing)**)**
2. **리눅스 스케줄러 전체 동작과 myrr 설명**
   1. **전체 스케줄러 동작**
      1. 어떤 프로세스의 task state가 running 상태로 전이되면 각 스케줄러가 가진 서브 런큐에 task를 enqueue 한다.
      2. schedule 함수가 호출되면 pick next task 함수 호출로 다음으로 CPU에서 실행될 프로세스를 선택하고 현재 실행중인 프로세스는 put prev task 함수 호출을 통해 다시 서브 run queue에 넣거나 종료된 경우 dequeue 한다.
      3. pick next task 함수로에서 선택된 프로세스가 이 후 CPU에서 실행되게 된다.
   2. **myrr 스케줄러 동작**
      1. RR 기반의 스케줄링을 위해 이전 과제와 마찬가지로 이중 원형 링크드 리스트를 사용한다.
      2. 인큐된 task는 head prev 즉 가장 뒤쪽으로 먼저 보낸다.
      3. 이 후 next task를 선택할 시 head next를 고르게 한다.
      4. 선택된 태스크가 일정 time slice 모두 소진하게 되면 현재의 task를 가장 뒤쪽으로 보내고 그 다음 task를 선택한다.
      5. 위의 과정을 반복하여 일정 시간 간격으로 현재 실행되는 task를 바꾸게 된다.
      6. 실행이 끝난 task는 dequeue된다.
3. **myrr.c 설명**
   1. **자료구조**
      1. 이중 원형 링크드 리스트
         1. 
         2. 양방향으로 연결이 되어있어서 노드 탐색이 양쪽으로 가능한 이중 링크드 리스트의 특징을 가진다.
         3. 원형 리스트로 헤드 노드와 테일 노드가 연결되어있다.
   2. **함수**
      1. update\_curr\_myrr
         1. 
         2. myrr entity의 업데이트 횟수를 1 증가
         3. 만약 업데이트 횟수가 정해진 time slice(여기서는 업데이트 횟수) 를 넘게 된다면 그 entity의 업데이트 횟수 변수를 초기화 한다.
         4. 그 후 현재의 프로세스를 리스트의 맨 뒤로 보내서 서브 런큐의 순서를 정렬하고 resched\_curr 함수를 통해 리스케줄링 작업을 진행한다.
      2. enqueue\_task\_myrr
         1. 
         2. Running 상태로 진입해 새로 들어온 프로세스의 구조체를 링크드 리스트 뒤쪽에 달아준다.
      3. dequeue\_task\_myrr
         1. 
         2. 현재 리스트에 들어와있는 프로세스 구조체가 있다면 하나를 초기화한 후 dequeue한다.
      4. pick\_next\_task\_myrr
         1. 
         2. 현재 서브런큐 내부에 하나라도 구조체가 존재한다면 pick next 작업을 진행한다.
         3. 헤드가 가리키고 있는 구조체를 찾은 후 리턴한다.
4. **최종 결과**
   1. 
   2. 결과 화면에서 자세히 보면 1-2-3-4-5->pick next 순서로만 출력이 된 것이 아니라 딱 맞아떨어지지 않게 작동하는 경우가 생기는 것을 확인할 수 있다
   3. 1879를 예로 말해보자면 위에서 먼저 1-2-3-4-5-1 로 실행되었고 그 결과로 아래에서도 2-3-4-5로 실행되게 된다.
   4. 5-pick이 아닌 5-1-pick이 되는 상황이 발생하게 되는 과정은 다음과 같다. update num 이 4에서 5가 되었을 때 time slice의 소진을 체크하는 조건문에 걸리게 되고 0 으로 초기화하는 코드까지 실행하였다고 하자. 그 상황에서 그 사이 시간이 흘러 타이머 인터럽트로 인해 원래 함수가 잠시 멈춘 상태로 다시 update curr myrr 함수가 실행된다. 현재 update num이 0 인 상태에서 1로 증가하게 되고 조건문에는 걸리지 않아 그대로 끝나게 된다. 이 후 중단되었던 함수의 작업을 마무리하게 되어 resched\_curr 함수가 실행되고 이 때 pick next 함수가 불리게 된다.