2주차 과제 보고서

cpu 스케줄러의 종류인 RR과 FCFS의 차이는 타이머 인터럽트의 여부다. 즉, RR에서 FCFS로 바꾸기 위해서는 타이머 인터럽트를 제거하여 하나의 프로세스가 exit 혹은 I/O버스트가 나오기 전까지 sched가 작동해서는 안된다. Sched가 존재하는 함수인 yield에서 프로세스의 state를 Runnable로 바꾸는 부분과 swtch함수를 주석처리를 하는 것도 일종의 방법이 될 수 있다. 하지만 이는 yield 함수 자체를 수정하는 것으로서 타이머 인터럽트 외의 상황에서 yield를 사용할 경우 제대로 작동될 수 없게 된다. 다음으로 생각해 볼 것은 타이머 인터럽트를 정지시키는 방법이다. 조정하기 위해서는 하드웨어/소프트웨어의 인터럽트를 정의하는 trap.c를 살펴봐야 한다. trap.c에서 약 165번째 줄을 보면 clockintr()라는 함수가 존재한다. 이 clockintr 함수는 tick을 이용해서 시간을 측정하는 함수다. 바로 밑에 devintr를 보면 clockintr로 시간을 측정하며 특정 시간에 도달하면 2를 반환하여 인터럽트를 일으키는 구조를 볼 수 있다. 여기서 반환되는 값인 2를 0으로 바꿔서 정상적으로 인터럽트를 일으키지 않을 수 있다. 또 다른 방법으로는 83번째 줄에 타임 인터럽트를 일으키는 코드가 주어져 있다. devintr의 반환값이 2라면 yield함수가 실행되며 그 yield함수 안에 sched가 작동되어 스케줄러로 바뀌게 된다. 다시 말해서 타임 인터럽트를 일으키는 if부분만 주석처리를 하면 타임 인터럽트가 일어나지 않게 된다. 위의 방법은 usertrap이며 kerneltrap에서 timeintr를 방지하기 위해서는 약 150번째 줄 근처에 있는 if문을 주석처리 한다.

Timeintr를 해결했다면 다음 남은 문제는 프로세스가 들어온 순서대로 실행이 되어야 한다는 것이다. RR에서는 들어온 순서가 아닌 RUNNABLE인 것을 기준으로 돌리기에 들어온 순서대로 프로세스가 실행 된다는 보장이 없다. 그렇기에 que를 새로 정의하고 que에 접근할 때 lock을 따로 설정해서 멀티코어가 동시에 접근하는 불상사를 없애야 한다. 또한 userinit 등의 정의된 함수에서 RUNNABLE로 바뀌는 곳들에서는 que에 새로 추가하는 명령어를 추가해야 한다. RUNNABLE이 될 경우 그 순서대로 que에 추가해야 하기에 addque라는 새로운 함수를 정의해서 RUNNABLE 바로 밑에 추가한다. 마지막으로 scheduler의 수정이다. Scheduler를 보면 for문으로 돌아다니며 RUNNABLE인 프로세스를 레지스터에 적재시키는 일을 한다. 여기서 RUNNABLE로 찾아다니는 부분을 주석으로 처리하고 그 부분에 que에서 순서대로 가져오며 스케줄러와 바꾸는 부분으로 새로 작성한다.

-권동휘