**OS (Operating System)**

**Second Term**

**작업환경: ubuntu-14.04.4-desktop-amd64,**

**VirtualBox-5.0.16-105871-Win**

**컴퓨터학과**

**2014210057 최준명**

**제출 일자: 2016.03.31**

1. **Guide Line**
2. **CFS는 어떠한 정책을 가지고 스케줄링 하는가?**  
   Completely Fair Scheduler(이하 CFS)는 multi-tasking processor에 기반한 알고리즘이고 각 프로세서들을 같은 시간 동안 공정하게 시간을 배분하여 작동시킨다.
3. **우선순위를 결정하는 기준은 무엇이고 어떻게 우선순위를 업데이트(재계산)하는가?**  
   우선순위는 그것의 스케쥴링을 하는 동안 vruntime의 증가가 얼마나 빠른지에 따라 결정된다. 이 스케쥴링으로 인한 시간차이는 작업의 우선순위 선정에 사용되는데 높은 우선순위의 작업의 Vruntime은 천천히 증가한다.

**B-1 Nice란 무엇인가?**

Nice란 작업이 할당받을 proportion of processor에 대한 weight로 사용하는 단위이다. 각각의 작업은 실행 가능한 작업들의 weight의 총 합을 기준으로 자신의 weight

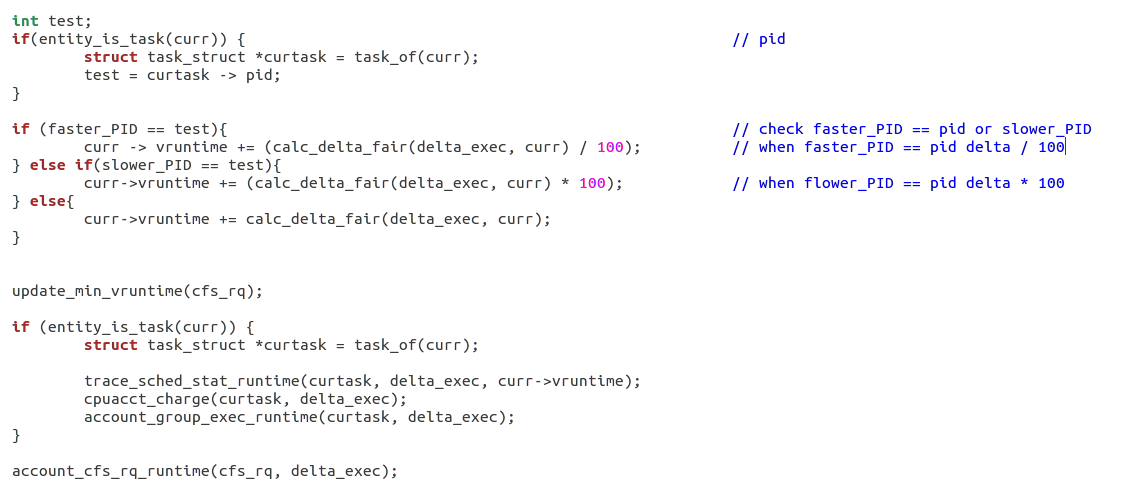
**B-2 Vruntime은 무엇인가?**

Vruntime(Virtual Runtime)은 normalized또는 weighted 된 작업들의 실제 실행된 시간을 나타낸다. Vruntime은 나노초 단위로 나타낸다.

**B-3 Vruntime은 어떻게 업데이트 되는가?**

Vruntime은 Scheduler\_tick()이 발생한 경우 update가 됩니다.

1. **스케줄러(do\_schedule())가 호출되는 3가지 시점은 언제인가?**
2. 인터럽트의 서비스 루틴이 종료되는시점
3. 현재 수행되고 있는 작업의 Tile Slice가 지났을 때
4. 새로 생성된 태스크가 현재 수행되고 있는 태스크를 선점해야 할 때
5. **태스크가 웨이크 업(wake up)되었을 때, 스케줄러는 어떠한 동작을 수행하는가?**  
   모든 Processes 에게 공평한 Processor을 나눠주려 한다. 이것은 인코더를 선점하고 text editor를 작동시킨다.
6. **Vruntime의 증가속도를 조절하는 것 이외에 특정 프로세스를 빠르게 스케줄링 할 수 있는 자신만의 아이디어를 제시하시오.**
7. **추가된 부분**
8. **Fair.c**

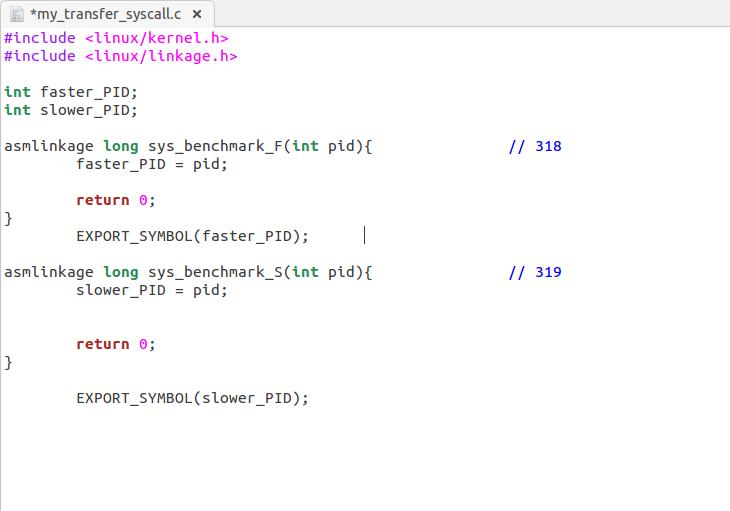


우선 curr의 Task형태가 존재하는지 확인하고 존재한다면 이것을 curtask로 꺼내고 여기서 pid를 꺼냈습니다.

그 다음 시스템콜에서 받은 faster\_PID, slower\_PID와 값과 비교하여 각 상황에 따라 vruntime의 증가량을 조절함으로써 각 프로세서에게 할당되는 vruntime이 다르게 조절하였습니다.

* 주석에 한글이 안쳐져서 제대로 된 설명은 하지 못했습니다.

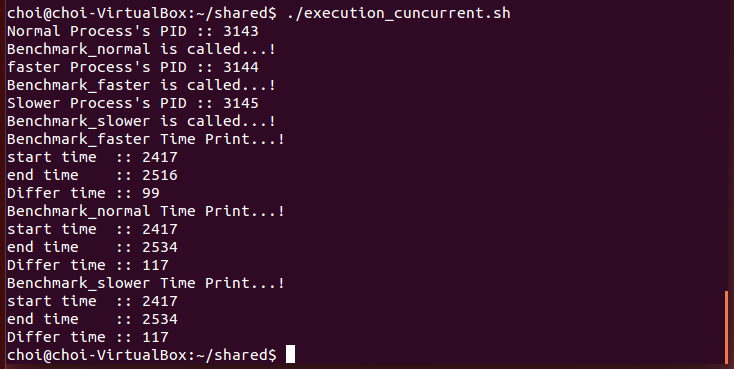
1. **My\_faster\_syscall.c**



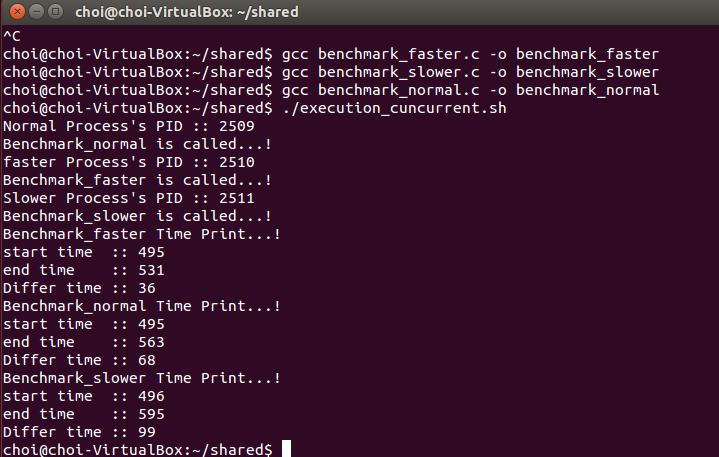
위의 사진은 faster\_PID, slower\_PID를 fair.c에서도 쓸 수 있도록 만들어 주기 위해 만든 system call 파일입니다. Benchmark 도중 pid를 syscall의 인자로 전달받아 faster\_PID, slower\_PID로 복사한 후 EXPORT\_SYMBOL로 모든 공간에서 사용할 수 있도록 만드는 역할을 합니다.

* 주석에 한글이 안쳐져서 제대로 된 설명은 하지 못했습니다.

1. **Result**
2. Scheduling 변환 전



1. Scheduling 변환 후



1. **결과분석**

변환 전과 변환 후의 값에서 PID의 값은 각 차이가 없지만 Differ time 즉 Delta 값은 우리가 위에서 수행해준 일들로 인하여 차이가 발생하였고 특정 상황에 따라 Vruntime의 증가 량에 차이를 만들었음을 알 수 있다. Benchmark\_faster에서는 Delta(Vruntime 증가량)를 줄여서 priority를 높이고 더 많이 접근할 수 있도록 해주고 benchmark\_slower에서는 Delta(Vruntime 증가량)을 늘려서 더 priority를 낮추고 덜 접근하도록 되었음을 알 수 있습니다.

1. **과제를 수행하면서 발생한 문제점 및 해결 방법**

이번 과제는 vruntime을 조정하기 위하여 benchmark\_faster, benchmark\_slower, benchmark\_normal에서 보내지는 각각의 pid를 구분하기 위하여 그냥 보내지는 pid이외에 faster에서는 faster\_PID, slower에서는 slower\_PID를 EXPORT\_SYMBOL을 이용하여 따로 보내주는 것이 첫번째 포인트였고 fair.c에서 vruntime이 update되는 부분을 찾는 것이 두번째 포인트 그리고 curr의 task를 이용하여 pid를 받아 faster\_PID, slower\_PID와 비교하여 Vruntime을 수정해주는 것이 마지막 포인트였습니다

첫번째 포인트에서 EXPORT\_SYMBOL 매크로를 함수 밖에 써야하는데 sys\_benchmark\_F, sys\_benchmark\_S함수안에 각각 매크로를 넣어서 오류가 생겼었습니다.

두번째 포인트에서는 PDF파일이 영어로된 문서여서 힘들었습니다. 해석이 잘되지 않아서 대략적인 내용은 이해가 됬었는데 정확한 해석이 안되서 문제가 있었습니다.

세번째 포인트는 정말 힘들었는데 pid를 어떻게 받아야하는지 감이 잡히지 않아서 정말 힘들었습니다.

1. **느낀점**

이번 과제를 하면서 OS가 CPU를 어떤 방식으로 배정하는지 얕게 나마 알 수 있었습니다. 처음에 생각했던 난이도 보다 좀 더 높았고 ‘조교’님의 높으신 아량이 없었다면 과제를 끝마칠 수 없었을 것입니다.