과제 #3 : xv6 SSU Scheduler

○ 과제 목표

- xv6의 프로세스 관리 및 스케줄링 기법 이해
- xv6의 기존 스케줄러를 FreeBSD 5.4 이후 구현된 ULE(non-interactive) 스케쥴러를 기반으로 하는 SSU 스케쥴러 및 테스트 프로그램 구현
  - ✓ 프로세스의 우선순위 + 프로세스가 사용한 CPU 시간을 스케줄링에 반영

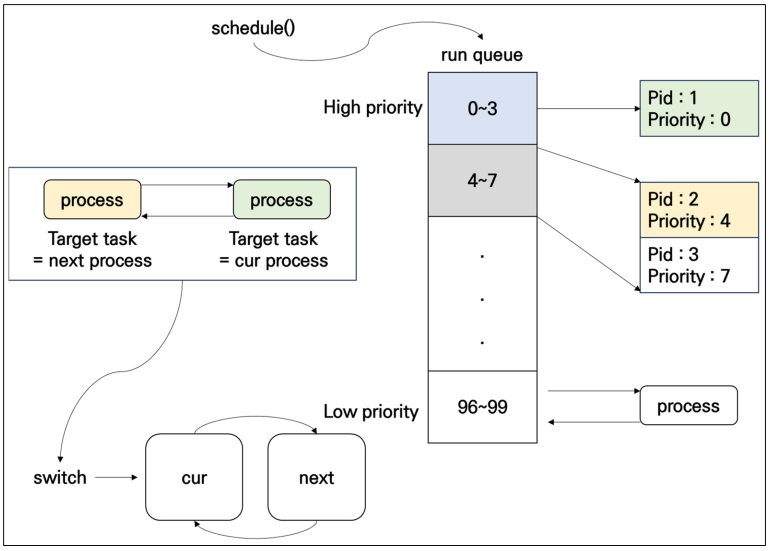
○ 기본 지식

- 스케줄링
  - ✓ 스케줄링은 다중 프로그래밍을 가능하게 하는 운영체제 커널의 기본 기능임
  - ✓ 기존 xv6의 스케줄링 기법은 다음 실행할 프로세스를 process table을 순회하며 RUNNABLE 상태인 프로세스를 순차적으로 선택함

○ 과제 내용

1. 기존 xv6 스케쥴러 분석 – 함수 단위로 상세하게 분석과 함수간 콜 그래프 등은 필수로 포함되어야 함.
   - ✓ 관련한 질문 받지 않음. 학생들이 창의적으로 생각해서 분석하기 바람.

2. SSU Scheduler 구현
   - ✓ 모든 프로세스는 CPU 사용량에 따라 우선순위(priority)를 가짐
   - ✓ 우선순위(priority)의 값이 낮을수록 우선순위가 높은 프로세스를 뜻함
   - ✓ 우선순위가 높은 프로세스부터 수행되기 때문에 기아 현상이 나타날 수 있음
   - ✓ time quantum(time slice)는 30tick으로 설정

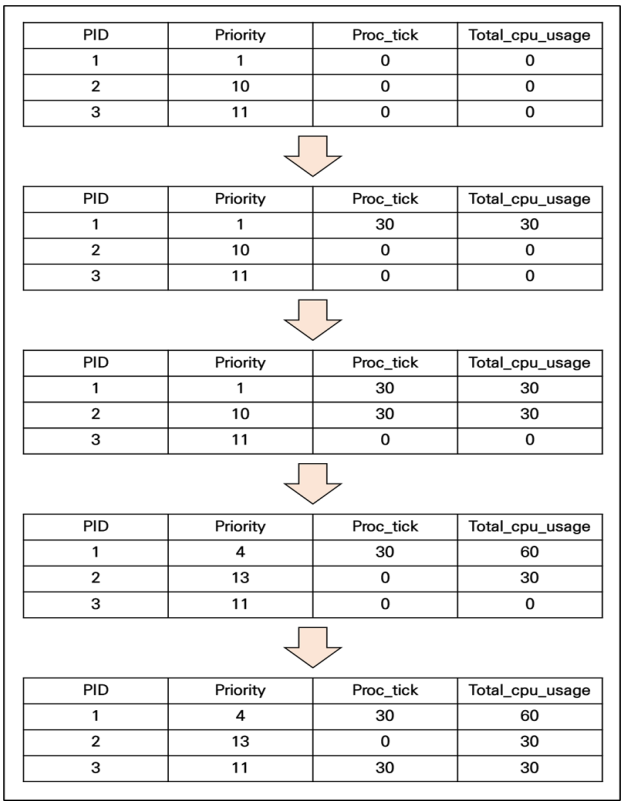


[그림 1] SSU 스케줄러의 동작 과정

   - ✓ 각 run queue는 우선순위 4개를 포함하고 있음
   - ✓ 현재 프로세스와 context switch 할 프로세스 탐색
     - run queues에서 우선순위가 높으며 비어있지 않은 큐 탐색
     - 위에서 찾은 큐를 기반으로 실행 가능 상태(RUNNABLE) 이면서 가장 낮은 priority 값을 가진 프로세스를 선택함
   - ✓ 모든 프로세스는 각자의 프로세스 아이디(PID), 상태(State) 및 우선순위(priority) 값을 가질때 클럭 tick 마다 proc_tick값을 증가시켜 CPU 사용량을 계산함
     - proc.h의 proc 구조체 내에 proc_tick 변수 추가 필요
   - ✓ priority 값은 ~~스케줄링 함수~~ 우선순위 재계산 함수가 호출될 때마다, 다음과 같은 규칙에 따라 업데이트됨
     - new_priority = old_priority + (~~proc_tick~~ priority_tick / 10)
     - 스케줄링은 60tick 마다 CPU 사용량에 따라 priority 값을 재계산함
     - 스케줄링되어 실행되는 프로세스들의 총 CPU 사용 시간(ticks)을 구해 60ticks 마다 우선순위 재조정 함수를 실행
   - ✓ 프로세스 생성 또는 wake up 시 priority 값을 0부터 부여하게 되면, 해당 프로세스가 독점 실행될 수 있으므로 run queue 내에서 관리하는 프로세스의 priority 값 중 가장 작은 값을 부여함(PID 1,2를 제외한 프로세스가 없을

시 priority의 값은 0을 가짐)
- xv6에서 프로세스 wake up은 프로세스 상태가“SLEEPING”에서 “RUNNABLE”로 전이되는 것을 의미함

3. 스케줄링 테스트를 할 수 있는 scheduler_test.c 프로그램 및 초기 priority 값을 설정할 수 있는 set_sche_info() 시스템콜 추가



PID	Priority	Proc_tick	Total_cpu_usage
1	1	0	0
2	10	0	0
3	11	0	0

PID	Priority	Proc_tick	Total_cpu_usage
1	1	30	30
2	10	0	0
3	11	0	0

PID	Priority	Proc_tick	Total_cpu_usage
1	1	30	30
2	10	30	30
3	11	0	0

PID	Priority	Proc_tick	Total_cpu_usage
1	4	30	60
2	13	0	30
3	11	0	0

PID	Priority	Proc_tick	Total_cpu_usage
1	4	30	60
2	13	0	30
3	11	30	30

~~[그림 2] SSU 스케줄러 동작 예시~~

✓ 해당 명령어 구현을 위해 필요한 매크로 선언은 다음과 같음
- “PNUM”
  ☞ 구현한 스케줄링 함수를 확인하기 위해 fork()로 생성할 프로세스의 개수
  ☞ 기본값 : 3

✓ PNUM 값을 변경해도 정상 동작하여야 함
✓ 해당 명령어는 생성한 프로세스가 모두 종료된 후 종료되도록 구현
✓ set_sche_info()는 프로세스의 초기 priority, 종료 타이머(tick)를 인자로 전달하여 스케줄링에 반영할 수 있어야함
- set_sche_info()을 통해 초기 priority 설정을 하지 않으면 현재 runq 내에서 관리하는 프로세스의 priority 값 중 가장 작은 값을 부여함(PID 1, 2 제외)

✓ 그림 2와 v0.9 명세서의 scheduler_test.c는 CPU 2개일 때 이상적인 결과, 본 과제는 CPU 1개일 때 측정하는 것이므로 아래 실행 결과를 따르길 바람.
✓ 코드 구현을 돕기 위해 예시 1-2, 1-3를 첨부하며 PID, priority, proc_tick, total_cpu_usage 출력문은 예시를 위해 time quantum 종료 시점, context switch 직전, 프로세스 종료 시점 총 3개로 각 1, 2, 3 으로 표시.

(예시 1-1). scheduler_test.c 실행 결과 set_sche_info(1, 110) ,set_sche_info (10, 60), set_sche_info(11, 60)

```
$ scheduler_test
start scheduler_test
set_sche_info() pid = 4 //set_sche_info 시스템콜 수행
PID : 4, priority : 1, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
set_sche_info() pid = 5 //set_sche_info 시스템콜 수행
PID : 5, priority : 12, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
set_sche_info() pid = 6 //set_sche_info 시스템콜 수행
PID : 6, priority : 11, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 4, priority : 6, proc_tick : 20 ticks, total_cpu_usage : 110 ticks (3)
PID : 4 terminated
PID : 5, priority : 12, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (3)
PID : 5 terminated
PID : 6, priority : 14, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (3)
PID : 6 terminated
end of scheduler_test
$
```

(예시 1-2). scheduler_test.c 실행 결과 set_sche_info(1, 110) ,set_sche_info (22, 200), set_sche_info(11, 250)

```
$ scheduler_test
PID : 2, priority : 99, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
start scheduler_test
PID : 4, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 4 //set_sche_info 시스템콜 수행
PID : 4, priority : 1, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 5, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 5 //set_sche_info 시스템콜 수행
PID : 5, priority : 25, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 6, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 6 //set_sche_info 시스템콜 수행
PID : 6, priority : 11, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 4, priority : 3, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 4, priority : 6, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 4, priority : 6, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 4, priority : 6, proc_tick : 20 ticks, total_cpu_usage : 110 ticks (3)
PID : 4 terminated
```

```
PID : 3, priority : 14, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 6, priority : 14, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 6, priority : 15, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 6, priority : 15, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 6, priority : 15, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 6, priority : 15, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 6, priority : 21, proc_tick : 30 ticks, total_cpu_usage : 120 ticks (1)
PID : 6, priority : 21, proc_tick : 0 ticks, total_cpu_usage : 120 ticks (2)
PID : 6, priority : 21, proc_tick : 30 ticks, total_cpu_usage : 150 ticks (1)
PID : 6, priority : 21, proc_tick : 0 ticks, total_cpu_usage : 150 ticks (2)
PID : 6, priority : 27, proc_tick : 30 ticks, total_cpu_usage : 180 ticks (1)
PID : 5, priority : 25, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 5, priority : 25, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 5, priority : 25, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 5, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 5, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 5, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 120 ticks (1)
PID : 5, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 120 ticks (2)
PID : 5, priority : 35, proc_tick : 30 ticks, total_cpu_usage : 150 ticks (1)
PID : 6, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 180 ticks (2)
PID : 6, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 210 ticks (1)
PID : 6, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 210 ticks (2)
PID : 6, priority : 33, proc_tick : 30 ticks, total_cpu_usage : 240 ticks (1)
PID : 6, priority : 33, proc_tick : 0 ticks, total_cpu_usage : 240 ticks (2)
PID : 6, priority : 33, proc_tick : 10 ticks, total_cpu_usage : 250 ticks (3)
PID : 6 terminated
PID : 3, priority : 37, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 5, priority : 37, proc_tick : 0 ticks, total_cpu_usage : 150 ticks (2)
PID : 5, priority : 40, proc_tick : 30 ticks, total_cpu_usage : 180 ticks (1)
PID : 5, priority : 40, proc_tick : 0 ticks, total_cpu_usage : 180 ticks (2)
PID : 5, priority : 40, proc_tick : 20 ticks, total_cpu_usage : 200 ticks (3)
PID : 5 terminated
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
end of scheduler_test
PID : 2, priority : 99, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
$
```

(예시 1-3). scheduler_test.c(출력문 추가) 실행 결과

set_sche_info(1, 300) ,set_sche_info (22, 600), set_sche_info(34, 600)

```
$ scheduler_test
PID : 2, priority : 99, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
start scheduler_test
```

```
PID : 4, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 4 //set_sche_info 시스템콜 수행
PID : 4, priority : 1, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 5, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 5 //set_sche_info 시스템콜 수행
PID : 5, priority : 24, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 6, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
set_sche_info() pid = 6 //set_sche_info 시스템콜 수행
PID : 6, priority : 34, proc_tick : 30 ticks, total_cpu_usage : 30 ticks (1)
PID : 4, priority : 4, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 4, priority : 6, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 4, priority : 6, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 4, priority : 6, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 4, priority : 12, proc_tick : 30 ticks, total_cpu_usage : 120 ticks (1)
PID : 4, priority : 12, proc_tick : 0 ticks, total_cpu_usage : 120 ticks (2)
PID : 4, priority : 12, proc_tick : 30 ticks, total_cpu_usage : 150 ticks (1)
PID : 4, priority : 12, proc_tick : 0 ticks, total_cpu_usage : 150 ticks (2)
PID : 4, priority : 18, proc_tick : 30 ticks, total_cpu_usage : 180 ticks (1)
PID : 4, priority : 18, proc_tick : 0 ticks, total_cpu_usage : 180 ticks (2)
PID : 4, priority : 18, proc_tick : 30 ticks, total_cpu_usage : 210 ticks (1)
PID : 4, priority : 18, proc_tick : 0 ticks, total_cpu_usage : 210 ticks (2)
PID : 4, priority : 24, proc_tick : 30 ticks, total_cpu_usage : 240 ticks (1)
PID : 5, priority : 24, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 5, priority : 24, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 5, priority : 24, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 5, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 4, priority : 24, proc_tick : 0 ticks, total_cpu_usage : 240 ticks (2)
PID : 4, priority : 24, proc_tick : 30 ticks, total_cpu_usage : 270 ticks (1)
PID : 4, priority : 24, proc_tick : 0 ticks, total_cpu_usage : 270 ticks (2)
PID : 4, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 300 ticks (3)
PID : 4 terminated
PID : 3, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
PID : 5, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 5, priority : 29, proc_tick : 30 ticks, total_cpu_usage : 120 ticks (1)
PID : 5, priority : 29, proc_tick : 0 ticks, total_cpu_usage : 120 ticks (2)
PID : 5, priority : 34, proc_tick : 30 ticks, total_cpu_usage : 150 ticks (1)
PID : 5, priority : 34, proc_tick : 0 ticks, total_cpu_usage : 150 ticks (2)
PID : 5, priority : 34, proc_tick : 30 ticks, total_cpu_usage : 180 ticks (1)
PID : 5, priority : 34, proc_tick : 0 ticks, total_cpu_usage : 180 ticks (2)
PID : 5, priority : 40, proc_tick : 30 ticks, total_cpu_usage : 210 ticks (1)
PID : 6, priority : 37, proc_tick : 0 ticks, total_cpu_usage : 30 ticks (2)
PID : 6, priority : 37, proc_tick : 30 ticks, total_cpu_usage : 60 ticks (1)
PID : 6, priority : 37, proc_tick : 0 ticks, total_cpu_usage : 60 ticks (2)
PID : 6, priority : 42, proc_tick : 30 ticks, total_cpu_usage : 90 ticks (1)
PID : 5, priority : 40, proc_tick : 0 ticks, total_cpu_usage : 210 ticks (2)
PID : 5, priority : 40, proc_tick : 30 ticks, total_cpu_usage : 240 ticks (1)
PID : 5, priority : 40, proc_tick : 0 ticks, total_cpu_usage : 240 ticks (2)
```

```
PID : 5, priority : 45, proc_tick : 30 ticks, total_cpu_usage : 270 ticks (1)
PID : 6, priority : 42, proc_tick : 0 ticks, total_cpu_usage : 90 ticks (2)
PID : 6, priority : 42, proc_tick : 30 ticks, total_cpu_usage : 120 ticks (1)
PID : 6, priority : 42, proc_tick : 0 ticks, total_cpu_usage : 120 ticks (2)
PID : 6, priority : 47, proc_tick : 30 ticks, total_cpu_usage : 150 ticks (1)
PID : 5, priority : 45, proc_tick : 0 ticks, total_cpu_usage : 270 ticks (2)
PID : 5, priority : 45, proc_tick : 30 ticks, total_cpu_usage : 300 ticks (1)
PID : 5, priority : 45, proc_tick : 0 ticks, total_cpu_usage : 300 ticks (2)
PID : 5, priority : 50, proc_tick : 30 ticks, total_cpu_usage : 330 ticks (1)
PID : 6, priority : 47, proc_tick : 0 ticks, total_cpu_usage : 150 ticks (2)
PID : 6, priority : 47, proc_tick : 30 ticks, total_cpu_usage : 180 ticks (1)
PID : 6, priority : 47, proc_tick : 0 ticks, total_cpu_usage : 180 ticks (2)
PID : 6, priority : 52, proc_tick : 30 ticks, total_cpu_usage : 210 ticks (1)
PID : 5, priority : 50, proc_tick : 0 ticks, total_cpu_usage : 330 ticks (2)
PID : 5, priority : 50, proc_tick : 30 ticks, total_cpu_usage : 360 ticks (1)
PID : 5, priority : 50, proc_tick : 0 ticks, total_cpu_usage : 360 ticks (2)
PID : 5, priority : 55, proc_tick : 30 ticks, total_cpu_usage : 390 ticks (1)
PID : 6, priority : 52, proc_tick : 0 ticks, total_cpu_usage : 210 ticks (2)
PID : 6, priority : 52, proc_tick : 30 ticks, total_cpu_usage : 240 ticks (1)
PID : 6, priority : 52, proc_tick : 0 ticks, total_cpu_usage : 240 ticks (2)
PID : 6, priority : 57, proc_tick : 30 ticks, total_cpu_usage : 270 ticks (1)
PID : 5, priority : 55, proc_tick : 0 ticks, total_cpu_usage : 390 ticks (2)
PID : 5, priority : 55, proc_tick : 30 ticks, total_cpu_usage : 420 ticks (1)
PID : 5, priority : 55, proc_tick : 0 ticks, total_cpu_usage : 420 ticks (2)
PID : 5, priority : 60, proc_tick : 30 ticks, total_cpu_usage : 450 ticks (1)
PID : 6, priority : 57, proc_tick : 0 ticks, total_cpu_usage : 270 ticks (2)
PID : 6, priority : 57, proc_tick : 30 ticks, total_cpu_usage : 300 ticks (1)
PID : 6, priority : 57, proc_tick : 0 ticks, total_cpu_usage : 300 ticks (2)
PID : 6, priority : 62, proc_tick : 30 ticks, total_cpu_usage : 330 ticks (1)
PID : 5, priority : 60, proc_tick : 0 ticks, total_cpu_usage : 450 ticks (2)
PID : 5, priority : 60, proc_tick : 30 ticks, total_cpu_usage : 480 ticks (1)
PID : 5, priority : 60, proc_tick : 0 ticks, total_cpu_usage : 480 ticks (2)
PID : 5, priority : 65, proc_tick : 30 ticks, total_cpu_usage : 510 ticks (1)
PID : 6, priority : 62, proc_tick : 0 ticks, total_cpu_usage : 330 ticks (2)
PID : 6, priority : 62, proc_tick : 30 ticks, total_cpu_usage : 360 ticks (1)
PID : 6, priority : 62, proc_tick : 0 ticks, total_cpu_usage : 360 ticks (2)
PID : 6, priority : 67, proc_tick : 30 ticks, total_cpu_usage : 390 ticks (1)
PID : 5, priority : 65, proc_tick : 0 ticks, total_cpu_usage : 510 ticks (2)
PID : 5, priority : 65, proc_tick : 30 ticks, total_cpu_usage : 540 ticks (1)
PID : 5, priority : 65, proc_tick : 0 ticks, total_cpu_usage : 540 ticks (2)
PID : 5, priority : 70, proc_tick : 30 ticks, total_cpu_usage : 570 ticks (1)
PID : 6, priority : 67, proc_tick : 0 ticks, total_cpu_usage : 390 ticks (2)
PID : 6, priority : 67, proc_tick : 30 ticks, total_cpu_usage : 420 ticks (1)
PID : 6, priority : 67, proc_tick : 0 ticks, total_cpu_usage : 420 ticks (2)
PID : 6, priority : 72, proc_tick : 30 ticks, total_cpu_usage : 450 ticks (1)
PID : 5, priority : 70, proc_tick : 0 ticks, total_cpu_usage : 570 ticks (2)
PID : 5, priority : 70, proc_tick : 30 ticks, total_cpu_usage : 600 ticks (3)
PID : 5 terminated
```

```
PID : 3, priority : 72, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
PID : 6, priority : 72, proc_tick : 0 ticks, total_cpu_usage : 450 ticks (2)
PID : 6, priority : 75, proc_tick : 30 ticks, total_cpu_usage : 480 ticks (1)
PID : 6, priority : 75, proc_tick : 0 ticks, total_cpu_usage : 480 ticks (2)
PID : 6, priority : 75, proc_tick : 30 ticks, total_cpu_usage : 510 ticks (1)
PID : 6, priority : 75, proc_tick : 0 ticks, total_cpu_usage : 510 ticks (2)
PID : 6, priority : 81, proc_tick : 30 ticks, total_cpu_usage : 540 ticks (1)
PID : 6, priority : 81, proc_tick : 0 ticks, total_cpu_usage : 540 ticks (2)
PID : 6, priority : 81, proc_tick : 30 ticks, total_cpu_usage : 570 ticks (1)
PID : 6, priority : 81, proc_tick : 0 ticks, total_cpu_usage : 570 ticks (2)
PID : 6, priority : 87, proc_tick : 30 ticks, total_cpu_usage : 600 ticks (3)
PID : 6 terminated
PID : 3, priority : 0, proc_tick : 0 ticks, total_cpu_usage : 1 ticks (2)
end of scheduler_test
PID : 2, priority : 99, proc_tick : 0 ticks, total_cpu_usage : 0 ticks (2)
$
```

(예시 2). scheduler_test.c

```
#include "types.h"
#include "stat.h"
#include "user.h"

void scheduler_func(void)
{
  // 구현
}

int main(void)
{
  scheduler_func();
  exit();
}
```

4. 다음에 실행될 프로세스 선정 과정 디버깅 기능 구현

✓ xv6 빌드 시 “debug=1”매개변수 전달을 통해, 스케줄링 함수에서 다음 실행될 프로세스를 선택할 때마다 현재 프로세스의 PID, 프로세스 이름, CPU 사용 시간(proc_tick), 프로세스 우선순위 값 등을 출력하도록 구현하되 학생들이 알아서 xv6와 SSU 스케쥴러를 비교할 수 있게 출력

✓ 다음 프로세스를 선택할 수 없는 경우에는 출력하지 않음

✓ 예시4, 예시 5는 debug 매개변수 및 디버그 콜에 따라 현재 프로세스의 pid와 이름을 출력하는 주요 코드로 참고 바람

(예시 3). xv6 빌드 시 “debug=1”매개변수 전달

```
$ make debug=1 qemu-nox
```

(예시 4). Makefile(아래 내용 추가 필요)

```
ifeq ($(debug), 1)
CFLAGS += -DDEBUG
endif
```

(예시 5). void scheduler(void)(예시 4와 연관)

```
#ifdef DEBUG
        if (p)
                cprintf("PID: %d, NAME: %s,\n", p->pid, p->name);
#endif
```

5. 위 1, 2, 3, 4를 기반으로 xv6 스케쥴러와 SSU 스케쥴러의 기능 및 성능 비교 분석
   - ✓ https://www.usenix.org/system/files/conference/atc18/atc18-bouron.pdf 참고하되 참고 논문과 같이 성능 분석을 제대로 할 필요는 없음.
   - ✓ 기능의 차이점과 성능의 차이점을 비교 분석하되, 기능의 차이는 도표를 포함해야 하고 성능의 차이는 반드시 그래프 포함해야 함
   - ✓ 관련한 질문 받지 않음. 학생들이 창의적으로 생각해서 비교 분석하기 바람.

○ 과제 주의 사항

1. 스케줄링에 사용되는 자료구조 및 변수
   - ✓ struct proc 구조체: proc.h 내에 선언되어있으며 프로세스에 대한 추가 정보 추적을 위해 변수 추가
     - priority: 우선순위 값으로 범위는 0~99이며, idle 프로세스는 99로 고정
     - proc_tick: 프로세스가 스케줄링 된 이후 다시 스케줄링 되기 전까지 CPU를 사용한 시간 (tick)으로 time quantum을 의미함
     - cpu used: 프로세스가 생성된 이후 CPU를 사용한 총합 시간(tick)
     - priority_tick: 우선순위 재조정 되기 전까지 CPU를 사용한 시간 (tick)
   - ✓ run queue 구현
     - run queue는 포인터 배열로 max index는 25를 가지고 있으며 프로세스간의 연결을 위해 링크드리스트로 구현
     - ~~각 연결리스트는 next, tail을 가지고 있음~~
     - run queues에서 프로세스가 삽입될 큐의 인덱스는 (프로세스의 priority / 4) 값임

2. priority 값 재계산
   - ✓ 매 클럭 tick 마다 proc_tick, cpu used 값을 증가시켜 CPU 사용량을 계산함
   - ✓ 스케줄링 된 이후 매 60tick 마다 CPU 사용량에 따라 priority값을 재계산함
   - ✓ priority 값을 변경한 후 run queues의 인덱스를 재계산하여 run queue 구조체의 tail을 통해 해당 인덱스 큐 맨 마지막에 삽입 (인덱스가 변경되지 않는 경우에도 큐의 맨 마지막에 삽입)

3. time quantum(or time slice) 수정
   - ✓ 기존 time quantum은 1 tick으로 설정되어 있음.

4. 필요한 변수 및 자료구조가 있으면 자유롭게 추가 가능

○ 기타 참고사항

- 해당 과제는 간단한 스케줄링 함수 구현이 목표이므로, CPU 코어 개수를 1개로 제한함
  - ✓ Makefile 수정 필요

○ 과제 제출 마감

- 2023년 11월 12일 (일) 23시 59분 59초까지 구글 클래스룸으로 제출
- 보고서 (hwp, doc, docx 등으로 작성)
- xv6에서 변경한 소스코드
- 마감시간 이후 24시간까지 지연 제출 가능. 그 이후 제출은 0점 처리. 설계과제 마감시간 이후 지연 제출은 30% 감점.

○ 필수 기능(구현)

- 2, 3, 4

○ 배점 기준

- 1 : 20점 (분석)
- 2 : 30점 (구현)
- 3 : 20점 (구현)
- 4 : 20점 (구현)
- 5 : 10점 (분석)