

<만든 변수에 대한 설명>

Runtime : 따로 정해진 정의가 없기 때문에 프로세스가 얼마만큼의 timeslice 를 할당받았는지 표시하는 변수로 사용했습니다. 자다 일어나는 프로세스의 경우에도 값이 초기화 되지 않고 계속 쌓입니다. 밀리틱 단위를 위해 1000 을 곱해서 출력합니다.

Vruntime : 받은 timeslice 에 weight 가 첨가된 가상의 값으로, 자다 일어났을 때는 특정한 값으로 초기화됩니다.

Total tick : trap.c 안의 ticks 라는 변수를 그대로 출력했습니다. 시간이 흐름에 따라 증가합니다. 역시나 밀리틱 단위를 위해 1000 을 곱해서 출력합니다.

<Testcfs.c 수정>

ps(); 코드를 ps(0);으로 수정해야 실행이 됩니다. ps 가 pid 라는 int 형 변수를 받아서 실행되기 때문입니다. 그리고 시간제한은 없지만, 연산 instruction 을 넣어놓은 for 문을 도는 횟수를 많이 줄여야 2 분 안에는 끝날 것 같습니다.

<코드 설명>

scheduler(): 다음으로 running 할 프로세스를 스케줄하는 함수입니다.

1. 최소 vruntime 을 가지고 있는 runnable 프로세스를 찾습니다. (Cpu 는 1 개인 것으로 가정함)(cpu 가 1 개이기 때문에 running process 도 runnable 에 포함시키라는 조건은 이해가 잘 되지 않지만 프로세스 timeslice 할당할 때 영향이 있도록 포함시켜 놓았습니다.)

2. 찾은 다음 그 프로세스의 timeslice 값을 알맞게 설정하고, runtime 에는 할당받은 timeslice 를 더해줍니다.

3. Vruntime 은 weight 에 따라 알맞은 값을 더해준 다음 그 프로세스는 running state 로 갑니다.

trap.c: 프로세스가 주어진 timeslice 를 다 쓰기 전에 종료가 되도 어차피 끝나는 프로세스이기 때문에 정확한 vruntime 은 의미가 없고, 마찬가지로 sleep 으로 가는 프로세스의 경우에도 일단은 runnable processes 에서 빠지기도 하고, 다시 wakeup 한다 해도 그 때는 vruntime 이 초기화되기 때문에 sleep 으로 갈 당시의 vruntime 은 의미가 없습니다. 그래서 trap.c 의 yield() 호출 부분에서 timeslice 동안 yield()를 하지 않고 잡아 두도록 했고, runtime 과 vruntime 은 미리 running state 로 가기 직전에 업데이트를 합니다.

allocproc(): 새로 들어온 프로세스에 대한 weight, runtime 등의 프로세스 구조체 변수 값을 초기화 해줍니다.

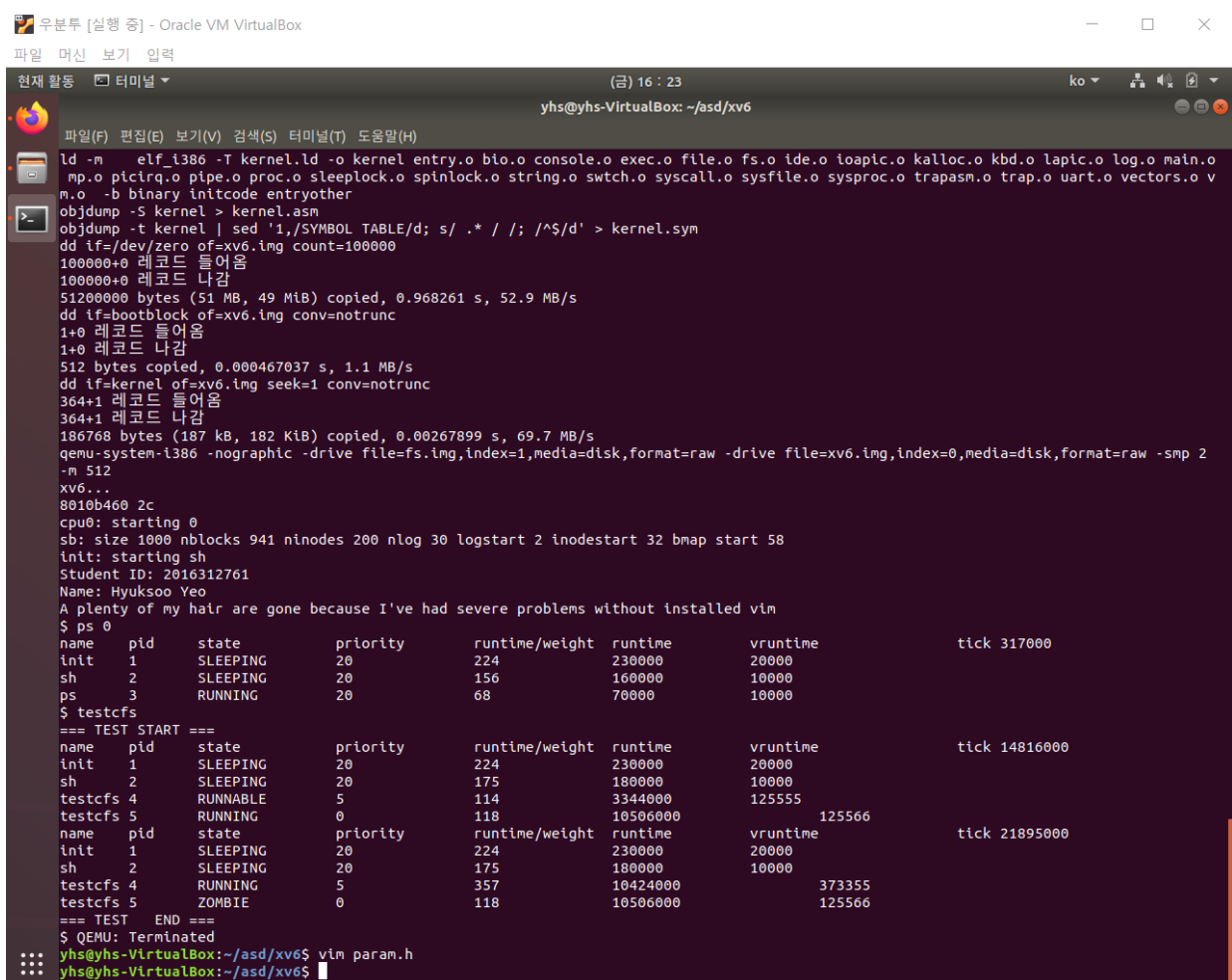
fork(): 자식 프로세스에게 vruntime 을 포함한 부모의 여러 변수 값을 그대로 옮겨줍니다.

wakeup1(): 자다 일어난 프로세스의 vruntime 을 초기화 하는데, 그 값은 현재 runnable 프로세스 중 최소 vruntime 값에 일어난 프로세스의 vruntime 으로 1 틱이 되는 값을 뺀 것이 됩니다.

<integer overflow>

console.c 의 %d 출력 알고리즘에서 sign bit 를 0 으로 바꿔서 unsigned int 범위까지 확장했습니다.

실행화면> max ncpu=1;



```
ld -m elf_i386 -T kernel.ld -o kernel entry.o bio.o console.o exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o main.o
mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o string.o swtch.o syscall.o sysfile.o sysproc.o trapasm.o trap.o uart.o vectors.o v
m.o -b binary initcode entryother
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=100000
100000+0 레코드 들어옴
100000+0 레코드 나감
51200000 bytes (51 MB, 49 MiB) copied, 0.968261 s, 52.9 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 레코드 들어옴
1+0 레코드 나감
512 bytes copied, 0.000467037 s, 1.1 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
364+1 레코드 들어옴
364+1 레코드 나감
186768 bytes (187 kB, 182 KiB) copied, 0.00267899 s, 69.7 MB/s
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2
-m 512
xv6...
8010b460 2c
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
Student ID: 2016312761
Name: Hyuksoo Yeo
A plenty of my hair are gone because I've had severe problems without installed vim
$ ps 0
  name  pid  state      priority    runtime/weight  runtime    vruntime    tick 317000
  init   1    SLEEPING    20          224/224         230000     20000
  sh     2    SLEEPING    20          156/156         160000     10000
  ps     3    RUNNING     20          68/68           70000      10000
$ testcfs
=== TEST START ===
  name  pid  state      priority    runtime/weight  runtime    vruntime    tick 14816000
  init   1    SLEEPING    20          224/224         230000     20000
  sh     2    SLEEPING    20          175/175         180000     10000
  testcfs 4    RUNNING     5          114/114         3344000    125555
  testcfs 5    RUNNING     0          118/118         10506000   125566
  name  pid  state      priority    runtime/weight  runtime    vruntime    tick 21895000
  init   1    SLEEPING    20          224/224         230000     20000
  sh     2    SLEEPING    20          175/175         180000     10000
  testcfs 4    RUNNING     5          357/357        10424000   373355
  testcfs 5    ZOMBIE      0          118/118         10506000   125566
=== TEST END ===
$ QEMU: Terminated
yhs@yhs-VirtualBox:~/asd/xv6$ vim param.h
yhs@yhs-VirtualBox:~/asd/xv6$
```