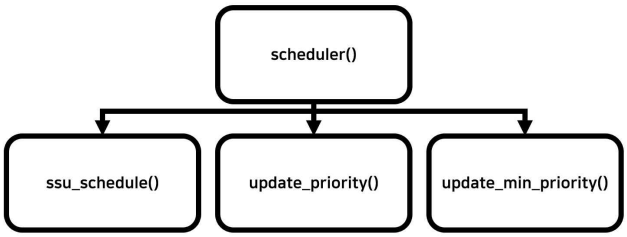
글로벌미디어학부 20182705 빙기범

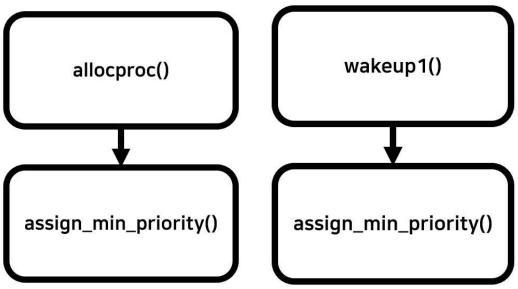
1. 개요

1) SSU Scheduler 구현
- 프로세스 실행 흐름에서 새로 생긴 프로세스가 제외되지 않도록 큰 가중치를 부여하는 스케줄러.
- 모든 프로세스는 각자의 가중치(weight) 및 우선순위(priority) 값을 가진다.
- 가장 작은 priority 값은 struct ptable의 멤버로 관리한다.
- time_slice = 10,000,000 ns
- CPU 코어 개수를 1개로 제한한다.
- init process가 처음 생기는 시점에서 최소 priority 값을 3으로 지정한다.

ssu_schedule() - RUNNABLE 상태인 프로세스 중 가장 낮은 priority 값을 가진 프로세스 리턴
update_priority() - 프로세스의 priority 값 업데이트(new_priority = old_priority + (time_slice / weight))
update_min_priority() - RUNNABLE 상태인 프로세스들의 priority 중 가장 낮은 priority 값 업데이트
assign_min_priority() - 프로세스의 priority에 가장 낮은 priority를



scheduler()
- 가장 낮은 priority 값을 가진 프로세스를 선택하여 다음에 실행하도록 함
- 실행한 프로세스의 priority 값을 업데이트
- 가장 낮은 priority 값을 업데이트



allocproc()
- 생성되는 child 프로세스의 weight, priority 값 부여

wakeup1()
- 프로세스 상태가 SLEEPING에서 RUNNABLE로 전이될 때, priority 값 부여

- 프로세스 생성 또는 wake up 시 priority 값을 0부터 부여하게 되면, 해당 프로세스가 독점 실행 될 수 있으므로 관리하고 있는 프로세스의 priority 값 중 가장 작은 값을 부여

2) 구현한 스케쥴링 함수의 동작 과정을 확인하는 sdebug 명령어 및 이를 위한 weightset() 시스템 콜 구현

- PNUM: 구현한 스케줄링 함수를 확인하기 위해 fork()로 생성할 프로세스의 개수

- PRINT_CYCLE:
* fork()로 생성된 프로세스 중 실행 중인 프로세스 정보를 출력하는 주기
* 정보는 프로세스 별로 한 번만 출력됨
* 출력 내용: 프로세스 ID, 프로세스 가중치, 프로세스 생성 후부터 정보가 출력되기까지 소요된 시간

- TOTAL_COUNTER:
* fork()로 생성된 프로세스가 생성된 이후부터 소모할 수 있는 시간
* 전부 소모하면 프로세스는 종료됨

- weightset(): 매개변수로 입력받은 값을 임의로 생성한 프로세스의 weight 값으로 부여
- sdebug 명령어에 의해 생성되는 프로세스의 weight 값은 시스템에서 생성한 프로세스와 별도로 생성된 순서에 따라 1부터 증가시키며 부여

- sdebug.c:
* 프로세스 개수만큼 for loop를 돌며 다음을 수행
1. 프로세스를 생성하고, 자식 프로세스인 경우, 조건에 따라 프로세스 정보 출력 및 종료 수행
2. 생성한 모든 자식 프로세스의 종료를 기다림

3) 다음에 실행될 프로세스 선정 과정 디버깅 기능 구현
- xv6 빌드 시 “debug=1” 매개변수 전달을 통해, 스케줄링 함수에서 다음 실행될 프로세스를 선택할 때마다 PID, 프로세스 이름, 프로세스 가중치 프로세스, 우선순위 값을 출력하도록 구현
- ssu_schedule()에 구현

2. 소스코드

1) Makefile

```
# 20182705
# DDEBUG 옵션 기능 구현
ifeq ($(debug),1)
CFLAGS += -DDEBUG
endif
```

sdebug 추가

```
UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_sdebug\
	_usertests\
	_wc\
	_zombie\
```

```
EXTRA=\
	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
	ln.c ls.c mkdir.c rm.c sdebug.c stressfs.c usertests.c wc.c zombie.c\
	printf.c umalloc.c\
	README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
	.gdbinit.tmpl gdbutil\
```

```
# 20182705
# CPU 최대 코어 수 변경
ifndef CPUS
CPUS := 1
endif
```

2) proc.c

```
//time_slice와 NULL 정의
#define TIME_SLICE 10000000
#define NULL ((void *)0)

//20182705
//weight 추가 및 초기화
int weight = 1;
```

```
struct {
  struct spinlock lock;
  struct proc proc[NPROC];
  //20182705
  //RUNNABLE 상태인 프로세스 중 가장 낮은 priority 값을 저장하는 min_priority 추가
  long long min_priority;
} ptable;
```

```
//20182705
//SSU Scheduler(): 가장 낮은 priority 값을 가진 프로세스 선택
struct proc*
ssu_schedule(void)
{
  struct proc *p;
  struct proc *ret = NULL;

  //ptable을 돌면서, RUNNABLE 상태인 프로세스들의 priority를 비교
  //가장 낮은 priority 값을 가진 프로세스를 리턴
  for (p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    if (p->state == RUNNABLE){
        if(ret == NULL || ret->priority > p->priority)
            ret = p;
    }
  }

  // DEBUG시 리턴될 프로세스를 선택하 때마다 PID, 프로세스 이름, 프로세스 가중치, 프로세스 우선순위 값을 출력
#ifdef DEBUG
  if (ret)
    cprintf("PID: %d, NAME: %s, WEIGHT: %d, PRIORITY: %d\n", ret->pid, ret->name, ret->weight, ret->priority);
#endif

  return ret;
}
```

```
60 //20182705
61 //update_priority(): 매개변수로 받은 프로세스의 priority 업데이트
62 void
63 update_priority(struct proc *proc)
64 {
65   //new_priority = old_priority + (time_slice / weight)의 규칙에 따라 업데이트)
66   proc->priority = proc->priority + (TIME_SLICE / proc->weight);
67 }
```

```
69 //20182705
70 //update_min_priority(): ptable의 min_priority를 RUNNABLE 상태인 프로세스들이 가진 priority 중 가장 작은 값으로 업데이트
71 void
72 update_min_priority()
73 {
74   struct proc *min = NULL;
75   struct proc *p;
76
77   //ptable을 돌면서 RUNNABLE 상태인 프로세스들의 priroity를 비교
78   //가장 낮은 priority 값을 가진 프로세스를 min에 기억
79   for (p = ptable.proc; p < &ptable.proc[NPROC]; p++){
80     if (p->state == RUNNABLE){
81         if(min == NULL || min->priority > p->priority)
82             min = p;
83     }
84   }
85
86   //min에 기억해둔 가장 낮은 priority 값으로 ptable의 min_priority 업데이트
87   if (min != NULL)
88     ptable.min_priority = min->priority;
89 }
```

```
91 //20182705
92 //assign_min_priority(): 매개변수로 받는 프로세스의 priority에 min_priority를 저장
93 void
94 assign_min_priority(struct proc *proc)
95 {
96   //프로세스의 priority에 ptable의 min_priority 저
97   proc->priority = ptable.min_priority;
98 }
```

```
151 allocproc(void)
152 {
153   struct proc *p;
154   char *sp;
155
156   acquire(&ptable.lock);
157
158   for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
159     if(p->state == UNUSED)
160       goto found;
161
162   release(&ptable.lock);
163   return 0;
164
165 found:
166   //20182705
167   //신규 생성되는 child 프로세스의 weight에 전역변수 weight값 할당
168   //전역변수 weight 1 증가
169   p->weight = weight++;
170   p->state = EMBRYO;
171   p->pid = nextpid++;
172
173   //20182705
174   //신규 생성되는 child 프로세스의 priority를 가장 작은 priority값 할당
175   assign_min_priority(p);
176
177   release(&ptable.lock);
```

```
205 void
206 userinit(void)
207 {
208   struct proc *p;
209   extern char _binary_initcode_start[], _binary_initcode_size[];
210
211   //20182705
212   //init process가 처음 생기는 시점에서 최소 priority 값을 3으로 지정
213   ptable.min_priority = 3;
214
215   p = allocproc();
216
```

```
412 scheduler(void)
413 {
414   struct proc *p;
415   struct cpu *c = mycpu();
416   c->proc = 0;
417
418   for(;;){
419     // Enable interrupts on this processor.
420     sti();
421
422     // Loop over process table looking for process to run.
423     acquire(&ptable.lock);
424
425     //20182705
426     //ssu_schedule()을 통해 RUNNABLE 상태의 프로세스 중 가장 낮은 priority를 가진 프로세스를 다음 실행되도록 선택
427     p = ssu_schedule();
428
429     if (p == NULL){
430       release(&ptable.lock);
431       continue;
432     }
433
434     // Switch to chosen process.  It is the process's job
435     // to release ptable.lock and then reacquire it
436     // before jumping back to us.
437     c->proc = p;
438     switchuvm(p);
439     p->state = RUNNING;
440
441     swtch(&(c->scheduler), p->context);
442     switchkvm();
443
444     //20182705
445     //선택되었던 프로세스의 priority를 규칙에 따라 업데이트
446     update_priority(p);
447     //RUNNABLE 상태의 프로세스 중 가장 낮으 priority 값을 업데이
448     update_min_priority();
449
450     // Process is done running for now.
451     // It should have changed its p->state before coming back.
452     c->proc = 0;
453
454     release(&ptable.lock);
455   }
456 }
```

```
557 static void
558 wakeup1(void *chan)
559 {
560   struct proc *p;
561
562   for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
563     if(p->state == SLEEPING && p->chan == chan){
564         //20182705
565         //프로세스 상태가 SLEEPING에서 RUNNABLE로 전이될 때, 해당 프로세스의 priority를 min_priority로 설정.
566         assign_min_priority(p);
567         p->state = RUNNABLE;
568     }
569 }
```

```
640 //20182705
641 //do_weightset(): 매개변수로 입력받은 값을 프로세스의 weight 값으로 부여
642 void
643 do_weightset(int weight)
644 {
645   //ptable lock을 얻고, 프로세스의 weight값을 부여하고, 다시 ptable lock을 품
646   acquire(&ptable.lock);
647   myproc()->weight = weight;
648   release(&ptable.lock);
649 }
```

3) proc.h

```
38 struct proc {
39   uint sz;                     // Size of process memory (bytes)
40   pde_t* pgdir;                // Page table
41   char *kstack;                // Bottom of kernel stack for this process
42   enum procstate state;        // Process state
43   int pid;                     // Process ID
44   struct proc *parent;         // Parent process
45   struct trapframe *tf;        // Trap frame for current syscall
46   struct context *context;     // swtch() here to run process
47   void *chan;                  // If non-zero, sleeping on chan
48   int killed;                  // If non-zero, have been killed
49   struct file *ofile[NOFILE];  // Open files
50   struct inode *cwd;           // Current directory
51   char name[16];               // Process name (debugging)
52   //20182705
53   //프로세스에 가중치, 우선순위 추가
54   int weight;                  // weight
55   long long priority;          // piriority
56 };
```

4) syscall.c

```
105 extern int sys_uptime(void);
106 //20182705
107 //sys_weightset 추가
108 extern int sys_weightset(void);
```

```
132 //20182705
133 //sys_weightset 추가
134 [SYS_weightset] sys_weightset,
135 };
```

5) syscall.h

```
22 #define SYS_close  21
23 //20182705
24 //sys_weightset 추가
25 #define SYS_weightset 22
```

6) sysproc.c

```
 93 //20182705
 94 //sys_weightset() 정의
 95 int
 96 sys_weightset(void)
 97 {
 98   int weight;
 99   //매개변수로 weight 값을 0보다 작거나 같은 값을 입력받을 시 에러처리
100   if (argint(0, &weight) <= 0)
101     return -1;
102
103   //weight 값 부여
104   do_weightset(weight);
105
106   return 0;
107 }
```

7) user.h

```
25 int uptime(void);
26 //20182705
27 //weightset() 추가
28 int weightset(int);
```

8) usys.S

```
31 SYSCALL(uptime)
32 //20182705
33 //weightset 추가
34 SYSCALL(weightset)
```

9) defs.h

```
104 //PAGEBREAK: 16
105 // proc.c
106 int             cpuid(void);
107 void            exit(void);
108 int             fork(void);
109 int             growproc(int);
110 int             kill(int);
111 struct cpu*     mycpu(void);
112 struct proc*    myproc();
113 void            pinit(void);
114 void            procdump(void);
115 void            scheduler(void) __attribute__((noreturn));
116 void            sched(void);
117 void            setproc(struct proc*);
118 void            sleep(void*, struct spinlock*);
119 void            userinit(void);
120 int             wait(void);
121 void            wakeup(void*);
122 void            yield(void);
123 //20182705
124 // do_wightset() 추가
125 void            do_weightset(int);
```

10) sdebug.c

```
 1 #include "types.h"
 2 #include "stat.h"
 3 #include "user.h"
 4
 5 #define PNUM 5
 6 #define PRINT_CYCLE 10000000
 7 #define TOTAL_COUNTER 500000000
 8
 9 void
10 sdebug_func(void)
11 {
12   int n, pid;
13   int start;
14   int counter;
15
16   printf(1, "start sdebug command\n");
17
18   //PNUM, 즉 fork()로 생성한 프로세스의 개수만큼 loop를 돎
19   for(n = 0; n < PNUM; n++){
20
21     //fork()로 프로세스를 생성하고, 생성한 시간을 start에 저장
22     pid = fork();
23     start = uptime();
24
25     //counter 0으로 초기화
26     counter = 0;
27     int print_done = 0;
28
29     if(pid < 0)
30       break;
31
32     //만약 생성된 프로세스가 자식 프로세스일 경우 if문안의 내용을 수행
33     if(pid == 0){
34
35       //생성된 프로세스의 weight값을 부여
36       //weight값은 n+1로, 생성된 순서에 따라 1부터 증가함
37       weightset(n+1);
38
39       //counter 값이 TOTAL_COUNTER이면 loop 종료 후 프로세스 종료
40       do{
41         //프로세스 정보가 한 번도 출력되지 않았고, counter값이 PRINT_CYCLE이면 프로세스 출력
42         if(print_done != 1 && counter == PRINT_CYCLE){
43           printf(1, "PID: %d, WEIGHT: %d, TIMES: %d ms\n", getpid(), n+1, (uptime()-start)*10);
44
45           //프로세스 정보가 출력됨을 알림
46           print_done = 1;
47         }
48         //프로세스가 수행될 때, counter 값은 clock_tick마다 1씩 증가
49         counter++;
50       }while(counter < TOTAL_COUNTER);
51
52       //프로세스 종료
53       printf(1, "PID: %d terminated\n", getpid());
54       exit();
55     }
56   }
57
58   //생성한 모든 자식 프로세스의 종료를 기다림
59   for(; n > 0; n--){
60     //자식 프로세스가 없는 경우 wait()은 -1을 반환, 종료
61     if(wait() < 0)
62       exit();
63   }
64   if(wait() != -1){
65     exit();
66   }
67
68   printf(1, "end sdebug command\n");
69 }
70
71 int main(void){
72   sdebug_func();
73   exit();
74 }
```

3. 결과

1) sdebug 명령어 실행 예시

```
$ sdebug
start sdebug command
PID: 8, WEIGHT: 5, TIMES: 120 ms
PID: 7, WEIGHT: 4, TIMES: 160 ms
PID: 6, WEIGHT: 3, TIMES: 240 ms
PID: 5, WEIGHT: 2, TIMES: 290 ms
PID: 4, WEIGHT: 1, TIMES: 700 ms
PID: 8 terminated
PID: 7 terminated
PID: 6 terminated
PID: 5 terminated
PID: 4 terminated
end sdebug command
$
```

2) 디버깅 기능 구현

```
$ kibum@kibum-VirtualBox:~/xv6-public$ make debug=1 qemu
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 1 -m 512
xv6...
cpu0: starting 0
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 10000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 10000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 30000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 40000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 40000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 50000003
init: stPID: 1, NAME: init, WEIGHT: 1, PRIORITY: 60000003
arting sh
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 70000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 80000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 90000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 70000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 75000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 75000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 80000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 80000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 85000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 90000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 95000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 95000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 100000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 100000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 105000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 105000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 110000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 110000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 115000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 115000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 115000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 120000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 120000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 125000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 125000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 125000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 130000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 130000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 135000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 135000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 140000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 140000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 145000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 145000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 150000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 155000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 160000003
$
```

3) 디버깅 및 sdebug 실행

(생략)

```
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 135000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 140000003
$ PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 145000003
sdebug
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 145000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 150000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 155000003
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 150000003
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 153333336
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 153333336
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 156666669
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 160000002
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 160000002
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 163333335
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 163333335
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 166666668
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 166666668
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 166666668
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 170000001
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 170000001
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 170000001
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 173333334
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 176666667
start sdebug command
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 180000000
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 183333333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 180000000
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 182500000
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 185000000
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 186666666
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 183333333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 185333333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 187333333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 187500000
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 189333333
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 189999999
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 186666666
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 188333332
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 189999998
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 190000000
```

(생략)

```
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 160333334
PID: 5, PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 160714284
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 161666664
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 161666666
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 158333333
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 159583333
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 160833333
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 162083333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 162142855
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 162333334
WEIGHT: 2, TIMES: 1PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 162500001
PID: 4, WEIGHT:PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 163333330
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 163333333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 163571426
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 164333334
80 ms
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 164583333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 164999996
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 164999997
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 164999999
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 165000001
 1, TIMES: 340PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 165833333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 166333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 166428568
PID: 7PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 166666662
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 167083333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 167500001
 PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 167857139
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 168333328
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 168333333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 168333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 169285710
, WEIGHT: 4, TIMES:PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 169285710
ms
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 169583333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 169999994
PID:PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 170333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 170714281
 280 ms
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 170833333
```

(생략)

```
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 172333334
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 173333333
: PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 173333333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 173333333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 173571423
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 174333334
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 174583333
8, WEIGHT: 5PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 174583333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 174583333
 6, WEIGHTPID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 174999994
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 175833333
, TIMES: 30PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 176249999
: 3, TIMES: 450PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 176333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 176428565
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 177083333
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 177083333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 177857136
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 177916665
 ms
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 177916665
0 ms
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 178333334
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 179166665
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 179285707
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 179583331
```

(생략

```
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 706249787
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 706333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 706428406
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 706666665
PID: 8 terminated
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 707083333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 707856977
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 707916453
PID: 8, NAME: sdebug, WEIGHT: 8, PRIORITY: 707916665
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 707916665
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 708333334
```

(생략)

```
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 756333334
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 756428391
PID: PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 757083333
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 757856962
7 terminated
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 757856962
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 757916433
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 758333334
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 759583099
```

(생략)

```
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 854333334
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 854583061
PID:PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 854583333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 856249727
 6 terminated
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 856333334
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 857083333
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 857916393
```

(생략)

```
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 968333334
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 969583333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 970333334
PID: 5 terminatedPID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 972083333
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 972333334

PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 972333334
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 974583333
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 975666667
```

(생략)

```
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1172083333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1174583333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1177083333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1179583333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1182083333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1184583333
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1187083333
PID: 4 tPID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 1189583333
erminated
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 1189583333
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 1192916666
end sdebug command
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 1196249999
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 1196249999
$
```