

1. 구현개요

모든 프로그램은 하위 디렉토리 안에 Makefile 이 존재한다 실행을 위해 make 로 만들어진 실행 파일을 이용하면 된다.

기본 요구사항에 더하여, RealTime FIFO 와 CPU_Burst_Time 을 모두 구현하였다.

코드는 cfs,cfsnice 를 첨부하고 커널 코드는 core.c(FIFO 적용, cpu_burst_time 출력) , stats.h(cpu_burst_timer 계산) , include/linux/sched.h(task_struct 에 변수추가) 를 첨부 하였다.

기본 연산과 데이터

기본 연산은 모두 행렬 덧셈이다. 같은 크기의 행렬을 더해 결과값을 저장하는 형태이다.

기본 데이터는 1024*1024 크기의 행렬로 선언하였다.

따라서 기본적으로 $1024 \times 1024 = 1,048,576$, 100 만 단위의 연산을 기본적으로 수행하고(for 문 을 위한 연산 제외) 각 연산을 반복하여 연산 수의 차이를 유도 하였다.

각 자식 프로세스의 연산 횟수 범위는

최소 100 번 반복 -> 1 억 번의 연산 ~ 2100 번 반복 -> 21 억 번의 연산으로 설정하였다

CFS 와 CFSnice 모두 같은 기본 연산을 다룬다.

출력 과정

두 프로그램 모두 구조체 배열, 공유 메모리를 이용하여 부모에서 출력해 준다.

공통적으로는 자신의 시작시간, 연산이 끝나는 시간. 연산 횟수를 가지고 있고, CFSnice 의 경우엔 할당 받은 nice 값이 들어가 있다. 끝나는 시간 순으로 정렬 후 출력해 준다. 자세한 결과는 아래 결과설명에서 다루겠다.

1-1 RealTime FIFO 를 위한 커널 코드 수정

전제

우선, 하는 방법을 설명하기 전에 Process 의 Scheduling 이 어떻게 이루어지는지 와 그때 필요한 자료구조를 간단히 설명하겠다.

리눅스 커널에선 Process 를 유연하게 scheduling 하기위해 scheduler 세부동작을 모듈화 하여 Scheduling Class 를 제공하고, Scheduling Class 는 각 Scheduler 별 세부 연산을 할 수 있게 Scheduling Class Method 를 제공한다.

모든 process 는 자신의 Task_Struct 안에 부모에서 물려받거나, 자신이 선택한 Scheduler Class 를 가리키기 위한 Scheduler Class Field 를 가지고 있다. 우선 순위에 따라 할당받는 Scheduler_Class 가 달라진다.

실패

Process 가 생성될 때 Scheduling Class 를 등록하기 위해 **sched_fork()** 함수에서 process 의 우선 순위에 따라 scheduling class 를 지정해 주는데 코드를 변경하였다.

```
3750
3751
3752     if (dl_prio(p->prio)){
3753         return -EAGAIN;
3754     }
3755     else if (rt_prio(p->prio)){
3756         p->sched_class = &rt_sched_class;
3757     }
3758     else if (p->prio > 120){
3759         p->sched_class = &rt_sched_class;
3760         p->policy = SCHED_FIFO;
3761     }else{
3762         p->sched_class = &fair_sched_class;
3763     }
3764
3765     init_entity_runnable_average(&p->se);
3766
3767     /*
```

위 코드처럼, 강제로 일부 process 를 rt_sched_class 로 변경하면 연관된 모든 코드를 수정하지 않는다면, 부팅이 되지 않는다. (부팅시 에러)

따라서 기본 적으로 정해주는 scheduling class 는 내버려 둔 후, 비슷한 효과를 보이도록 각 process 에 적용되는 Policy 와, priority 를 수정하는 작업을 수행했다. 과정은 아래와 같다.

할당 방법은 단계에 따라 3 단계 있다.

1. 우선 process 가 생성될 때 호출되는 **sched_fork()** 함수에서 fork 때 기본 설정을 하는 과정에서 Policy 를 SCHED_FIFO 로 설정하여 생성되는 모든 프로세스가 FIFO 방식으로 처리되게 유도하였다.

```
if (unlikely(p->sched_reset_on_fork)) {
    if (task_has_dl_policy(p) || task_has_rt_policy(p)) {
        p->policy = SCHED_FIFO;
        p->static_prio = NICE_TO_PRIO(0);
        p->rt_priority = 0;
    } else if (PRIO_TO_NICE(p->static_prio) < 0)
        p->static_prio = NICE_TO_PRIO(0);

    p->prio = p->normal_prio = __normal_prio(p);
    set_load_weight(p, false);

    /*
     * We don't need the reset flag anymore after the fork. It
     * fulfilled its duty:
     */
    p->sched_reset_on_fork = 0;
}
p->policy = SCHED_FIFO;
```

2. 1. 또한 스레드의 RT priority 와 policy 가 변경되는 **sched_setscheduler** 함수에서 역시 sched_policy 를 SCHED_FIFO 로 할당하고, SCHED_NORMAL 인 PROCESS 의 우선순위를 대략 40($NICE_WIDTH = 19 - (-20) + 1$) 정도 낮추어 주었다.

```
static int sched_setscheduler(struct task_struct *p, int policy,
                             const struct sched_param *param, bool check)
{
    struct sched_attr attr = {
        .sched_policy = SCHED_FIFO,
        .sched_priority = param->sched_priority,
        .sched_nice = PRIO_TO_NICE(p->static_prio),
    };
    if (attr.sched_policy == SCHED_NORMAL) {
        attr.sched_priority = param->sched_priority - NICE_WIDTH - attr.sched_nice;
        attr.sched_policy = SCHED_FIFO;
    }

    /* Fixup the legacy SCHED_RESET_ON_FORK hack. */
    if ((policy != SETPARAM_POLICY) && (policy & SCHED_RESET_ON_FORK)) {
        attr.sched_flags |= SCHED_FLAG_RESET_ON_FORK;
        policy &= ~SCHED_RESET_ON_FORK;
        attr.sched_policy = SCHED_FIFO;
    }
}
```

3. 1.+2. 에 추가로 **sched_fork()** 함수에서 역시 SCHED_NORAML 인 process 들의 우선순위를 낮추는 작업을 추가하였다. 일반 프로세스의 우선순위를 강제로 낮춰 아래에 있는 우선순위에 따라 scheduling class 를 지정하는 코드에서 RT_sched_class 가 할당되도록 유도했다.

```
/*
 * Revert to default priority/policy on fork if requested.
 */
if(p->policy == SCHED_NORMAL){
    p->prio = current->normal_prio - NICE_WIDTH - PRIO_TO_NICE(current->static_prio);
    p->normal_prio = p->prio;
    p->rt_priority = p->prio;
    p->policy = SCHED_FIFO;
    p->static_prio = NICE_TO_PRIO(0);
}
```

여러 실험을 해본 결과

1. 만 변경한 커널에선 Process 가 SCHED_OTHER 로 할당되었고.

1.+2. 까지 한 커널에선 SCHED_FIFO 로 동작 하긴 하지만 수행을 찍어보면 완벽하진 않다.

1+2+3 까지 수행한 커널에선 우선, 전체 시스템의 속도가 현저하게 느려졌으며 실험 코드에서 fork 한 process 가 FIFO 기반으로 동작하는걸 볼 수 있다.

(자세한 사항은 수행결과 설명)

1-2 CPU-Burst time 측정을 위한 커널 코드 수정

CPU-Burst time 측정을 위해선 Process 가 RunQueue 에서 CPU 에 할당 (hit) 되는 시점 시간, CPU 를 떠나는(depart) 시간이 필요하다.

/kernel/sched/cputime.c 에서 User Time, System Time, 등등을 구해 Task_struct 안 utime,stime 에 더해주는 함수가 있지만. 정확한 CPU-Burst time 은 cpu 에 들어갔다 나오는 시간을 구해야 한다 생각하여.

/kernel/sched/stats.h 에 있는 함수를 이용하여 값을 구하였다.

함수 호출 순서 : sched_info_arrive() (cpu 에 할당 되었을 때) -> sched_info_depart() (CPU 를 떠날 때)

sched_info 구조체에는 스케줄링 동작 세부정보가 저장되어 있다.

이 구조체 멤버 중 .last_arrival 변수엔 Process 가 Runqueue 에서 CPU 에 hit 되는 순간에 시간이 저장되어 있다. sched_info_arrive()함수는 re_cloack() 함수를 이용하여 현재 시간을 구한 후 last_arrival 를 업데이트 해준다.

Process 가 CPU 를 떠날 때 호출되는 **sched_info_depart()** 함수에서 delta 값을 이용하여 최종 Cpu_burst time 을 구하였다. + task_struc 안에 cpu_burst_time 변수를 선언 하였다.

```

33  */
34  static inline void sched_info_depart(struct rq *rq, struct task_struct *t)
35  {
36      unsigned long long delta = rq_clock(rq) - t->sched_info.last_arrival;
37      t->cpu_burst_time += delta;
38      rq_sched_info_depart(rq, delta);
39
40      if (t->state == TASK_RUNNING)
41          sched_info_queued(rq, t);
42  }
43
44  /*
45  * Called when tasks are switched involuntarily due, typically, to expiring

```

원래 동작은 rq 구조체에 delta 값을 누적하여 각 CPU 별 cpu_time 을 저장하는 것인데 Process 별 CPU-burst time 을 위해 task_struct 에 cpu_burst_time 변수를 추가하여 누적시켰다.

cpu_burst_time 의 초기화는 `/kernel/sched/core.c sched_fork()`에서 이루어 진다.

Process 상태가 task_struct 에 TASK_NEW 로 저장될 때를 이용하였다.

```

708  */
709  int sched_fork(unsigned long clone_flags, struct task_struct *p)
710  {
711      unsigned long flags;
712
713      __sched_fork(clone_flags, p);
714      /*
715       * We mark the process as NEW here. This guarantees that
716       * nobody will actually run it, and a signal or other external
717       * event cannot wake it up and insert it on the runqueue either.
718       */
719      p->state = TASK_NEW;
720      p->cpu_burst_time = 0;
721
722      /*

```

cpu_burst_time 의 출력은 프로세스가 종료 상태로 바뀌고 (TASK_DEAD) 최종 context switch 과정에서 이루어진다. 프로세스 종료과정을 따라가보면 `do_task_dead()` -> `context_switch()` -> `finish_task_dead()` 이다.

`do_task_dead()` 함수에서 TASK_DEAD 로 process 의 상태가 바뀌고

`Context switch()` 함수에서 새로운 Process 로 Context Switch 가 이루어 진후,

`finish_task_switch()`에서 새로운 Process 는 이전 Process 진짜로 TASK_DEAD 일때 그 process 의 Stack 공간을 해제 해주는데 이 타이밍을 이용하여 cpu_burst_time 을 출력해 주었다.

```

}
if (unlikely(prev_state == TASK_DEAD)) {
    printk("process %d cpu_burst_time %lld\n", prev->pid, prev->cpu_burst_time);
    if (prev->sched_class->task_dead)
        prev->sched_class->task_dead(prev);

    /*
     * Remove function-return probe instances associated with this
     * task and put them back on the free list.
     */
    kprobe_flush_task(prev);

    /* Task is done with its stack. */
    put_task_stack(prev);

    put_task_struct_rcu_user(prev);
}

```

2. 출력 결과 설명

CFS 적용 - 일을 같게 줄 때 (연산단위 : 10 억번, 정렬 : 빨리 끝나는 순)

```

junic@junic-VirtualBox:~/os/task4/cfs$ ./CFS
process 2188 work : 100 end at 03:37.080299 start at 03:35.436247 total time : 19644052
process 2183 work : 100 end at 03:37.101744 start at 03:35.431997 total time : 19669747
process 2173 work : 100 end at 03:37.162717 start at 03:35.424996 total time : 19737721
process 2189 work : 100 end at 03:37.179548 start at 03:35.435878 total time : 19743670
process 2185 work : 100 end at 03:37.213261 start at 03:35.437819 total time : 19775442
process 2193 work : 100 end at 03:37.223547 start at 03:35.437866 total time : 19785681
process 2180 work : 100 end at 03:37.243869 start at 03:35.431880 total time : 19811989
process 2192 work : 100 end at 03:37.256081 start at 03:35.444477 total time : 19811604
process 2191 work : 100 end at 03:37.258213 start at 03:35.440115 total time : 19818098
process 2182 work : 100 end at 03:37.261644 start at 03:35.432006 total time : 19829638
process 2190 work : 100 end at 03:37.269923 start at 03:35.437818 total time : 19832105
process 2184 work : 100 end at 03:37.275979 start at 03:35.431917 total time : 19844062
process 2179 work : 100 end at 03:37.276263 start at 03:35.426900 total time : 19849363
process 2178 work : 100 end at 03:37.280626 start at 03:35.435910 total time : 19844716
process 2177 work : 100 end at 03:37.282721 start at 03:35.425350 total time : 19857371
process 2174 work : 100 end at 03:37.285982 start at 03:35.425088 total time : 19860894
process 2181 work : 100 end at 03:37.290852 start at 03:35.431972 total time : 19858880
process 2176 work : 100 end at 03:36.954534 start at 03:35.425263 total time : 10529271
process 2186 work : 100 end at 03:37.291184 start at 03:35.435900 total time : 19855284
process 2187 work : 100 end at 03:37.298456 start at 03:35.436014 total time : 19862442
process 2175 work : 100 end at 03:37.309781 start at 03:35.425206 total time : 19884575
end

```

모두 거의 같은 시간에 시작하고 같은 시간에 끝난다.

CFS 적용 - 일을 다르게 줄 때 (연산 단위 : 100 = 10 억번, 정렬 : 빨리 끝나는 순)

연산이 많은 프로세스순으로 생성되도록 유도했다. (for 문 앞에 배치)

```
process 2341 work : 100 end at 12:30.438096 start at 12:28.412644 total time : 2025ms
process 2340 work : 200 end at 12:32.269566 start at 12:28.424657 total time : 3845ms
process 2339 work : 300 end at 12:33.711717 start at 12:28.431988 total time : 5279ms
process 2338 work : 400 end at 12:35.429604 start at 12:28.421515 total time : 7008ms
process 2337 work : 500 end at 12:37.009791 start at 12:28.421366 total time : 8589ms
process 2336 work : 600 end at 12:38.560766 start at 12:28.425193 total time : 10135ms
process 2335 work : 700 end at 12:39.172052 start at 12:28.419809 total time : 10753ms
process 2334 work : 800 end at 12:40.436267 start at 12:28.424812 total time : 12011ms
process 2333 work : 900 end at 12:41.194133 start at 12:28.419781 total time : 12775ms
process 2332 work : 1000 end at 12:43.127494 start at 12:28.419764 total time : 14708ms
process 2331 work : 1100 end at 12:43.709536 start at 12:28.419763 total time : 15289ms
process 2330 work : 1200 end at 12:44.298093 start at 12:28.419765 total time : 15879ms
process 2329 work : 1300 end at 12:45.043097 start at 12:28.415847 total time : 16628ms
process 2328 work : 1400 end at 12:47.161353 start at 12:28.419763 total time : 18742ms
process 2327 work : 1500 end at 12:46.577267 start at 12:28.424755 total time : 18152ms
process 2326 work : 1600 end at 12:46.826154 start at 12:28.431998 total time : 18394ms
process 2325 work : 1700 end at 12:47.378311 start at 12:28.410946 total time : 18968ms
process 2324 work : 1800 end at 12:48.233053 start at 12:28.410814 total time : 19823ms
process 2321 work : 2100 end at 12:48.416835 start at 12:28.410603 total time : 20006ms
process 2323 work : 1900 end at 12:48.656590 start at 12:28.410792 total time : 20245ms
process 2322 work : 2000 end at 12:48.768021 start at 12:28.410688 total time : 20357ms
end
```

생성 자체는 거의 비슷하게 되었지만, 끝나는 순서는 작업이 빨리 끝나는 순으로 끝난다.

CFSnice - 똑 같은 일의 양 + 빨리 끝나는 작업 (연산 단위 : 백만번)

```
process 2512 work : 1 nice : 19 end at 23:44.296783 start at 23:44.199942 total time : 96ms
process 2513 work : 1 nice : 19 end at 23:44.297829 start at 23:44.193719 total time : 104ms
process 2523 work : 1 nice : -20 end at 23:44.303416 start at 23:44.196459 total time : 106ms
process 2519 work : 1 nice : 0 end at 23:44.303454 start at 23:44.203796 total time : 99ms
process 2518 work : 1 nice : 0 end at 23:44.306113 start at 23:44.193365 total time : 112ms
process 2516 work : 1 nice : 0 end at 23:44.309765 start at 23:44.192924 total time : 116ms
process 2517 work : 1 nice : 0 end at 23:44.309785 start at 23:44.193249 total time : 116ms
process 2522 work : 1 nice : -20 end at 23:44.311242 start at 23:44.199995 total time : 111ms
process 2524 work : 1 nice : -20 end at 23:44.314795 start at 23:44.207685 total time : 107ms
process 2525 work : 1 nice : -20 end at 23:44.317419 start at 23:44.203654 total time : 113ms
process 2526 work : 1 nice : -20 end at 23:44.334110 start at 23:44.207694 total time : 126ms
process 2515 work : 1 nice : 0 end at 23:44.335137 start at 23:44.193143 total time : 141ms
process 2521 work : 1 nice : -20 end at 23:44.339547 start at 23:44.199939 total time : 139ms
process 2527 work : 1 nice : -20 end at 23:44.340993 start at 23:44.213858 total time : 127ms
process 2514 work : 1 nice : 0 end at 23:44.343227 start at 23:44.193035 total time : 150ms
process 2511 work : 1 nice : 19 end at 23:44.346696 start at 23:44.192173 total time : 154ms
process 2510 work : 1 nice : 19 end at 23:44.349795 start at 23:44.192084 total time : 157ms
process 2520 work : 1 nice : 0 end at 23:44.354030 start at 23:44.199948 total time : 154ms
process 2509 work : 1 nice : 19 end at 23:44.354304 start at 23:44.191972 total time : 162ms
process 2508 work : 1 nice : 19 end at 23:44.363366 start at 23:44.191786 total time : 171ms
process 2507 work : 1 nice : 19 end at 23:44.368590 start at 23:44.191699 total time : 176ms
```

생성 시간, 종료시간이 같다면 우선순위는 유의미한 차이 없음

CFSnice – 똑 같은 일의 양 + 오래 걸리는 작업 (연산 단위 : 십억번)

```
junie@junie-ubuntu:~/os/task4/cfsnice$ ./cfsnice
process 2571 work : 1000 nice : -20 end at 25:34.238556 start at 25:21.644164 total time : 12595ms
process 2569 work : 1000 nice : -20 end at 25:33.246709 start at 25:21.634647 total time : 11613ms
process 2570 work : 1000 nice : -20 end at 25:34.274851 start at 25:21.635914 total time : 12639ms
process 2568 work : 1000 nice : -20 end at 25:32.832645 start at 25:21.636555 total time : 11196ms
process 2561 work : 1000 nice : 0 end at 25:33.317851 start at 25:21.632006 total time : 11686ms
process 2563 work : 1000 nice : 0 end at 25:33.417777 start at 25:21.625030 total time : 11793ms
process 2567 work : 1000 nice : -20 end at 25:33.476762 start at 25:21.625347 total time : 11852ms
process 2565 work : 1000 nice : -20 end at 25:34.395665 start at 25:21.625185 total time : 12771ms
process 2564 work : 1000 nice : 0 end at 25:33.504371 start at 25:21.625097 total time : 11880ms
process 2562 work : 1000 nice : 0 end at 25:33.670847 start at 25:21.631980 total time : 12038ms
process 2566 work : 1000 nice : -20 end at 25:33.784982 start at 25:21.625265 total time : 12159ms
process 2560 work : 1000 nice : 0 end at 25:33.810580 start at 25:21.628158 total time : 12182ms
process 2558 work : 1000 nice : 0 end at 25:33.817046 start at 25:21.631993 total time : 12185ms
process 2559 work : 1000 nice : 0 end at 25:33.927988 start at 25:21.632017 total time : 12295ms
process 2554 work : 1000 nice : 19 end at 25:40.005274 start at 25:21.624352 total time : 18381ms
process 2551 work : 1000 nice : 19 end at 25:40.042708 start at 25:21.624144 total time : 18419ms
process 2553 work : 1000 nice : 19 end at 25:40.047272 start at 25:21.624264 total time : 18424ms
process 2555 work : 1000 nice : 19 end at 25:40.062564 start at 25:21.624438 total time : 18439ms
process 2552 work : 1000 nice : 19 end at 25:40.134083 start at 25:21.624226 total time : 18510ms
process 2557 work : 1000 nice : 19 end at 25:40.189656 start at 25:21.625642 total time : 18565ms
process 2556 work : 1000 nice : 19 end at 25:40.234822 start at 25:21.628615 total time : 18607ms
```

Nice 값이 19 인건 확실히 밀렸으나, 0 과 -20 은 약간 어긋나는 모습을 보임.

→ 연산의 량이 많을수록 우선순위가 중요해짐.

CFSnice – 다른 일의 양(우선순위가 낮을수록 적게함)+ 적게 걸리는 작업 (연산 단위 : 100 = 1 억번)

우선 순위가 낮을수록 먼저 생성, 적은 연산 유도함. 최고연산 5 억, 최저연산 1 억

```
process 2866 work : 300 nice : 0 end at 38:04.393552 start at 38:00.827877 total time : 3566ms
process 2864 work : 300 nice : 0 end at 38:04.542792 start at 38:00.826273 total time : 3717ms
process 2865 work : 300 nice : 0 end at 38:04.678494 start at 38:00.827902 total time : 3851ms
process 2860 work : 300 nice : 0 end at 38:04.682111 start at 38:00.825919 total time : 3857ms
process 2863 work : 300 nice : 0 end at 38:04.708646 start at 38:00.826084 total time : 3883ms
process 2867 work : 500 nice : -20 end at 38:06.020526 start at 38:00.831690 total time : 5189ms
process 2868 work : 500 nice : -20 end at 38:06.046420 start at 38:00.827885 total time : 5219ms
process 2862 work : 300 nice : 0 end at 38:04.764221 start at 38:00.826181 total time : 3939ms
process 2871 work : 500 nice : -20 end at 38:05.774906 start at 38:00.837509 total time : 4938ms
process 2870 work : 500 nice : -20 end at 38:05.836977 start at 38:00.835075 total time : 5001ms
process 2872 work : 500 nice : -20 end at 38:05.839136 start at 38:00.837489 total time : 5001ms
process 2873 work : 500 nice : -20 end at 38:05.842578 start at 38:00.843926 total time : 4999ms
process 2861 work : 300 nice : 0 end at 38:04.844570 start at 38:00.825991 total time : 4018ms
process 2869 work : 500 nice : -20 end at 38:05.993178 start at 38:00.827914 total time : 5165ms
process 2855 work : 100 nice : 19 end at 38:06.313258 start at 38:00.825086 total time : 5489ms
process 2857 work : 100 nice : 19 end at 38:06.386894 start at 38:00.825304 total time : 5562ms
process 2859 work : 100 nice : 19 end at 38:06.428521 start at 38:00.826598 total time : 5602ms
process 2858 work : 100 nice : 19 end at 38:06.445682 start at 38:00.827871 total time : 5618ms
process 2856 work : 100 nice : 19 end at 38:06.475262 start at 38:00.825171 total time : 5651ms
process 2854 work : 100 nice : 19 end at 38:06.514424 start at 38:00.825036 total time : 5690ms
process 2853 work : 100 nice : 19 end at 38:06.517764 start at 38:00.824936 total time : 5693ms
```


CFSnice – 다른 일의 양+ 오래 걸리는 작업 (연산 단위 : 1000 = 10 억번)

우선 순위가 낮을수록 먼저 생성, 적은 연산 유도함. 최고연산 50 억, 최저연산 10 억

```
junitc@junitc-VirtualBox:~/os/task4/cfsnice$ ./CFSnice
process 2645 work : 3000 nice : 0 end at 29:18.073690 start at 28:42.461732 total time : 35612ms
process 2649 work : 3000 nice : 0 end at 29:23.337163 start at 28:42.464225 total time : 40873ms
process 2655 work : 5000 nice : -20 end at 29:36.100328 start at 28:42.479825 total time : 53621ms
process 2643 work : 3000 nice : 0 end at 29:17.669425 start at 28:42.461480 total time : 35207ms
process 2648 work : 3000 nice : 0 end at 29:17.708108 start at 28:42.464253 total time : 35243ms
process 2647 work : 3000 nice : 0 end at 29:17.773961 start at 28:42.461953 total time : 35312ms
process 2656 work : 5000 nice : -20 end at 29:33.463627 start at 28:42.471602 total time : 50993ms
process 2650 work : 5000 nice : -20 end at 29:34.446158 start at 28:42.464233 total time : 51982ms
process 2644 work : 3000 nice : 0 end at 29:19.925960 start at 28:42.461597 total time : 37464ms
process 2652 work : 5000 nice : -20 end at 29:33.817295 start at 28:42.464253 total time : 51353ms
process 2651 work : 5000 nice : -20 end at 29:33.892371 start at 28:42.464235 total time : 51428ms
process 2654 work : 5000 nice : -20 end at 29:35.543960 start at 28:42.479826 total time : 53064ms
process 2646 work : 3000 nice : 0 end at 29:21.955213 start at 28:42.461846 total time : 39493ms
process 2653 work : 5000 nice : -20 end at 29:34.987561 start at 28:42.479825 total time : 52507ms
process 2641 work : 1000 nice : 19 end at 29:41.359576 start at 28:42.468084 total time : 58892ms
process 2640 work : 1000 nice : 19 end at 29:41.385558 start at 28:42.460726 total time : 58925ms
process 2637 work : 1000 nice : 19 end at 29:41.512737 start at 28:42.460511 total time : 59052ms
process 2642 work : 1000 nice : 19 end at 29:41.599782 start at 28:42.462319 total time : 59137ms
process 2639 work : 1000 nice : 19 end at 29:41.670333 start at 28:42.460676 total time : 59209ms
process 2638 work : 1000 nice : 19 end at 29:41.716955 start at 28:42.460597 total time : 59256ms
process 2636 work : 1000 nice : 19 end at 29:40.954739 start at 28:42.460436 total time : 58494ms
end
```

우선순위가 낮은 Process 는 대체로 먼저 생성 되었으며 연산수가 적어도 확실히 뒤로 밀렸리는 모습을 보임 , 하지만 0,-20 섞이는 모습을 보임.

CFSnice – 확실한 차이의 일의 양 (연산 단위 : 100 = 1 억번)

우선 순위가 낮을수록 먼저 생성 유도함. 최고연산 10 억, 최저연산 1 억

```
junitc@junitc-VirtualBox:~/os/task4/cfsnice$ ./CFSnice
process 2901 work : 800 nice : 0 end at 40:29.280863 start at 40:20.392226 total time : 8889ms
process 2909 work : 1000 nice : -20 end at 40:31.009003 start at 40:20.411710 total time : 10598ms
process 2910 work : 1000 nice : -20 end at 40:31.023679 start at 40:20.400203 total time : 10624ms
process 2911 work : 1000 nice : -20 end at 40:31.185975 start at 40:20.405039 total time : 10781ms
process 2908 work : 1000 nice : -20 end at 40:31.276449 start at 40:20.403725 total time : 10873ms
process 2904 work : 800 nice : 0 end at 40:30.307771 start at 40:20.400164 total time : 9908ms
process 2907 work : 1000 nice : -20 end at 40:31.287928 start at 40:20.400101 total time : 10888ms
process 2900 work : 800 nice : 0 end at 40:28.324476 start at 40:20.392539 total time : 7932ms
process 2903 work : 800 nice : 0 end at 40:29.487925 start at 40:20.392796 total time : 9095ms
process 2899 work : 800 nice : 0 end at 40:30.484311 start at 40:20.392446 total time : 10091ms
process 2906 work : 1000 nice : -20 end at 40:30.746462 start at 40:20.405049 total time : 10341ms
process 2905 work : 800 nice : 0 end at 40:30.752912 start at 40:20.400100 total time : 10352ms
process 2902 work : 800 nice : 0 end at 40:29.967713 start at 40:20.392709 total time : 9575ms
process 2912 work : 1000 nice : -20 end at 40:30.997553 start at 40:20.411711 total time : 10585ms
process 2894 work : 100 nice : 19 end at 40:31.508388 start at 40:20.391627 total time : 11116ms
process 2897 work : 100 nice : 19 end at 40:31.524702 start at 40:20.395896 total time : 11128ms
process 2893 work : 100 nice : 19 end at 40:31.535942 start at 40:20.391539 total time : 11144ms
process 2892 work : 100 nice : 19 end at 40:31.547373 start at 40:20.391396 total time : 11155ms
process 2898 work : 100 nice : 19 end at 40:31.580991 start at 40:20.393060 total time : 11187ms
process 2896 work : 100 nice : 19 end at 40:31.628337 start at 40:20.391776 total time : 11236ms
process 2895 work : 100 nice : 19 end at 40:31.641925 start at 40:20.391721 total time : 11250ms
end
```

우선 순위가 낮은 process 는 비교적 먼저 생성 되고 일이 적더라도(-20 의 10 배 적음) 확실히 늦게끝난다.

- nice 값 0,-20 이면 연산의 량이 비슷하면서 오래 걸릴 수록 그룹이 확실히 나뉜다.
- nice 값이 19 이면 확인결과 최대 10 배 적게 해도 늦게 끝난다.

CFS vs CFSnice

공통점 : 적은 일을 할수록 순서는 랜덤하고, 연산이 많아 질수록 연산 자체 시간이 중요하다.

차이점 : nice 값이 19 이면 확실히 우선순위가 낮은 process 는 밀린다.

CFS CPU – burst -time 측정

끝나는 시간순 정렬, 연산단위 100 = 1 억.

```

78 end
79 junic@junic-VirtualBox:~/os/task4/cfsnice$ cd ..
80 junic@junic-VirtualBox:~/os/task4$ cd cfs
81 junic@junic-VirtualBox:~/os/task4/cfs$ vim cfs.c
82 junic@junic-VirtualBox:~/os/task4/cfs$ ./CFS
83 process 2976 work : 100 end at 47:43.503953 start at 47:41.291914 total time : 2212ms [ 2836.745509] process 2976 cpu_burst_time 572483226
84 process 2975 work : 200 end at 47:44.837797 start at 47:41.295678 total time : 3542ms [ 2836.996567] process 2977 cpu_burst_time 18537998
85 process 2974 work : 300 end at 47:46.765398 start at 47:41.288801 total time : 5476ms [ 2838.079326] process 2975 cpu_burst_time 1116564160
86 process 2973 work : 400 end at 47:48.339677 start at 47:41.295691 total time : 7043ms [ 2840.006944] process 2974 cpu_burst_time 1501739567
87 process 2972 work : 500 end at 47:49.151763 start at 47:41.297089 total time : 7855ms [ 2841.596899] process 2973 cpu_burst_time 2084095838
88 process 2971 work : 600 end at 47:50.527839 start at 47:41.299364 total time : 9228ms [ 2842.393550] process 2972 cpu_burst_time 2627225059
89 process 2970 work : 700 end at 47:52.197047 start at 47:41.308053 total time : 10889ms [ 2842.603630] process 2931 cpu_burst_time 2221096
90 process 2969 work : 800 end at 47:52.391870 start at 47:41.295722 total time : 11095ms [ 2842.785064] process 2938 cpu_burst_time 516821
91 process 2968 work : 900 end at 47:54.967692 start at 47:41.295708 total time : 13671ms [ 2843.785169] process 2971 cpu_burst_time 3094877321
92 process 2967 work : 1000 end at 47:55.103072 start at 47:41.299331 total time : 13804ms [ 2844.288182] process 2942 cpu_burst_time 273457
93 process 2966 work : 1100 end at 47:55.923466 start at 47:41.316348 total time : 14607ms [ 2844.481225] process 2947 cpu_burst_time 246786
94 process 2965 work : 1200 end at 47:57.049559 start at 47:41.297228 total time : 15753ms [ 2845.438829] process 2970 cpu_burst_time 3587359523
95 process 2964 work : 1300 end at 47:59.012925 start at 47:41.295679 total time : 17718ms [ 2845.632664] process 2969 cpu_burst_time 4166877059
96 process 2963 work : 1400 end at 47:58.779242 start at 47:41.295678 total time : 17483ms [ 2848.209495] process 2968 cpu_burst_time 4680351596
97 process 2962 work : 1500 end at 47:59.346521 start at 47:41.295731 total time : 18050ms [ 2848.344912] process 2967 cpu_burst_time 4857515721
98 process 2961 work : 1600 end at 47:59.745142 start at 47:41.316346 total time : 18428ms [ 2849.165059] process 2966 cpu_burst_time 5356426707
99 process 2960 work : 1700 end at 47:59.784546 start at 47:41.287236 total time : 18497ms [ 2850.291116] process 2965 cpu_burst_time 6065266773
100 process 2959 work : 1800 end at 48:00.217621 start at 47:41.287096 total time : 18931ms [ 2852.021105] process 2963 cpu_burst_time 6885012020
101 process 2958 work : 1900 end at 48:00.734686 start at 47:41.286972 total time : 19447ms [ 2852.254517] process 2964 cpu_burst_time 6641671370
102 process 2957 work : 2000 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2852.588251] process 2962 cpu_burst_time 7767044393
103 process 2956 work : 2100 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2852.986916] process 2961 cpu_burst_time 8072751205
104 process 2955 work : 2200 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2853.026237] process 2960 cpu_burst_time 8551382620
105 process 2954 work : 2300 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2853.459206] process 2959 cpu_burst_time 8823953455
106 process 2953 work : 2400 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2853.976155] process 2958 cpu_burst_time 9264888188
107 process 2952 work : 2500 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2854.273240] process 2956 cpu_burst_time 9543742983
108 process 2951 work : 2600 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2854.600108] process 2957 cpu_burst_time 9013849942
109 process 2950 work : 2700 end at 48:01.031807 start at 47:41.286763 total time : 19746ms [ 2854.600518] process 2955 cpu_burst_time 3626823
110 end
junic@junic-VirtualBox:~/os/task4/cfs$
junic@junic-VirtualBox:~/os/task4/cfs$

```

연산 결과 정리

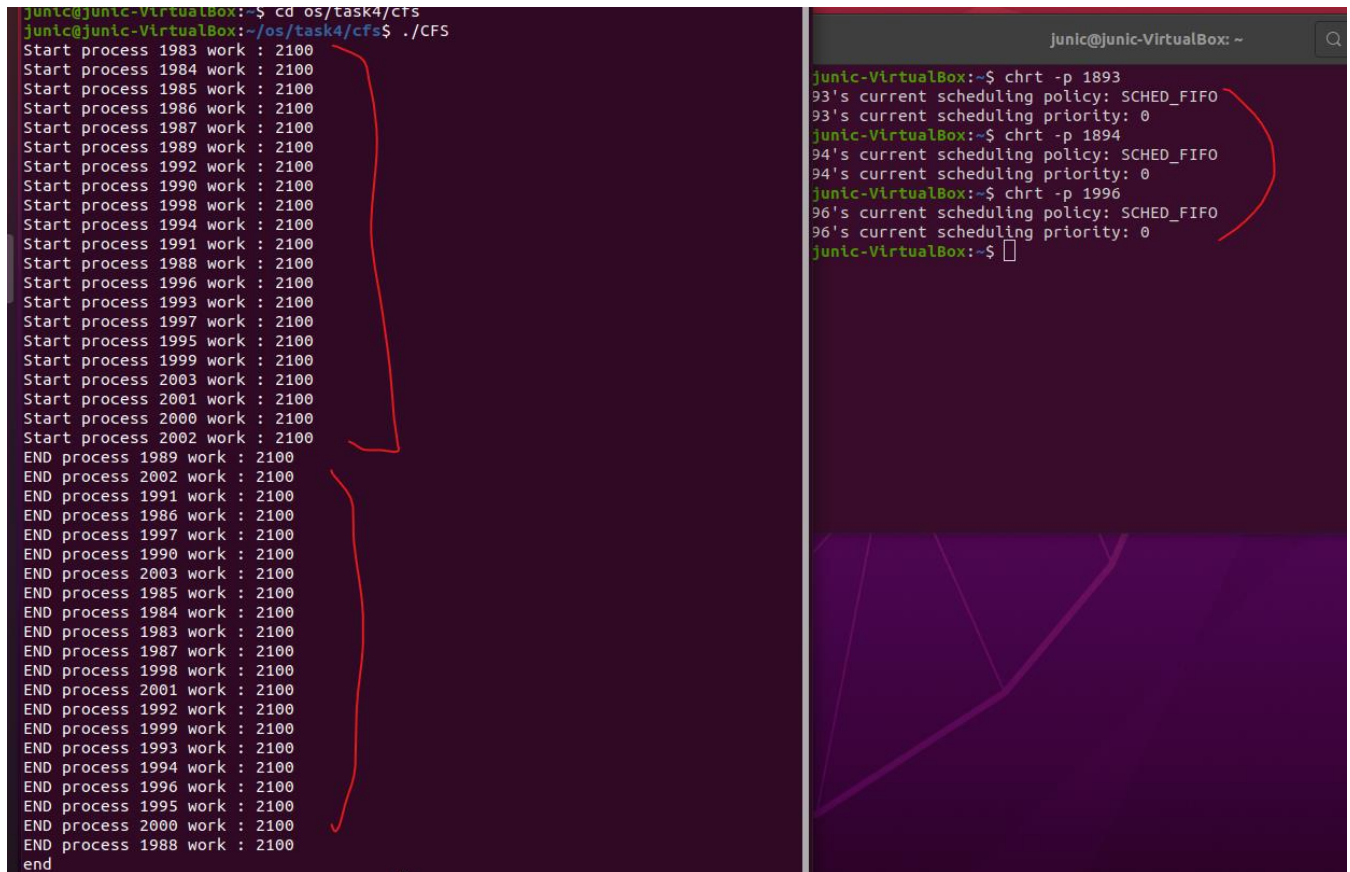
연산(백만)	총 시간(ms)	cpu_burst_time(ms)	비율
100	2212	572.48	0.258807
200	3542	1116.56	0.315234
300	5478	1501.74	0.27414
400	7043	2084.1	0.295911
500	7855	2672.77	0.340264
600	9228	3094.88	0.335379
700	10889	3587.36	0.329448
800	11095	4166.88	0.375564
900	13671	4680.35	0.342356
1000	13804	4857.52	0.351892
1100	14607	5356.43	0.366703
1200	15753	6065.27	0.385023
1300	17718	6641.67	0.374854
1400	17483	6885.01	0.393812
1500	18505	7769.04	0.419835
1600	18428	8072.75	0.43807
1700	18497	8823.95	0.477048
1800	18931	8823.95	0.466111
1900	19746	9264.89	0.469203
2000	20071	9543.74	0.475499
2100	19746	9013.85	0.45649

대략적인 비율로 볼 때 연산 수가 많을수록 cpu 사용 비율이 높아진다.

RealTime-FIFO 스케줄링

1. 번까지 변경한 커널로 수행해본 결과는 다른 수행결과와 같으므로 생략.

1+2 번까지 변경한 커널로 수행해본 결과.



```
junic@junic-VirtualBox:~$ cd os/task4/cfs
junic@junic-VirtualBox:~/os/task4/cfs$ ./CFS
Start process 1983 work : 2100
Start process 1984 work : 2100
Start process 1985 work : 2100
Start process 1986 work : 2100
Start process 1987 work : 2100
Start process 1989 work : 2100
Start process 1992 work : 2100
Start process 1990 work : 2100
Start process 1998 work : 2100
Start process 1994 work : 2100
Start process 1991 work : 2100
Start process 1988 work : 2100
Start process 1996 work : 2100
Start process 1993 work : 2100
Start process 1997 work : 2100
Start process 1995 work : 2100
Start process 1999 work : 2100
Start process 2003 work : 2100
Start process 2001 work : 2100
Start process 2000 work : 2100
Start process 2002 work : 2100
END process 1989 work : 2100
END process 2002 work : 2100
END process 1991 work : 2100
END process 1986 work : 2100
END process 1997 work : 2100
END process 1990 work : 2100
END process 2003 work : 2100
END process 1985 work : 2100
END process 1984 work : 2100
END process 1983 work : 2100
END process 1987 work : 2100
END process 1998 work : 2100
END process 2001 work : 2100
END process 1992 work : 2100
END process 1999 work : 2100
END process 1993 work : 2100
END process 1994 work : 2100
END process 1996 work : 2100
END process 1995 work : 2100
END process 2000 work : 2100
END process 1988 work : 2100
end

junic-VirtualBox:~$ chrt -p 1893
93's current scheduling policy: SCHED_FIFO
93's current scheduling priority: 0
junic-VirtualBox:~$ chrt -p 1894
94's current scheduling policy: SCHED_FIFO
94's current scheduling priority: 0
junic-VirtualBox:~$ chrt -p 1996
96's current scheduling policy: SCHED_FIFO
96's current scheduling priority: 0
junic-VirtualBox:~$
```

chrt -p 명령어로는 SCHED_FIFO 로 동작 한다. 하지만 전체적인 수행 결과는 CFS 와 비슷하다.

1+2+3 번까지 변경한 커널로 수행해본 결과.

이 버전의 커널에선 전체 시스템 느려짐. + 멀티 테스킹이 안된다.

터미널 역시 하나의 터미널이 수행중이면 다른 터미널 이용불가여서 chrt 명령어로 확인하지 못했다.

서로 다른 연산 수

```
junic@junic-VirtualBox:~/os/task4/cfs$ ./CFS
Start process 2172 work : 2100
Start process 2173 work : 2000
Start process 2174 work : 1900
Start process 2175 work : 1800
Start process 2176 work : 1700
Start process 2177 work : 1600
END process 2177 work : 1600
Start process 2178 work : 1500
END process 2176 work : 1700
Start process 2179 work : 1400
END process 2174 work : 1900
Start process 2180 work : 1300
END process 2175 work : 1800
Start process 2181 work : 1200
END process 2172 work : 2100
Start process 2182 work : 1100
END process 2173 work : 2000
Start process 2183 work : 1000
END process 2181 work : 1200
END process 2182 work : 1100
Start process 2184 work : 900
Start process 2185 work : 800
END process 2180 work : 1300
Start process 2186 work : 700
END process 2178 work : 1500
Start process 2187 work : 600
END process 2183 work : 1000
Start process 2188 work : 500
END process 2179 work : 1400
Start process 2189 work : 400
END process 2189 work : 400
Start process 2190 work : 300
END process 2188 work : 500
Start process 2191 work : 200
END process 2186 work : 700
Start process 2192 work : 100
END process 2187 work : 600

END process 2192 work : 100
END process 2185 work : 800
END process 2191 work : 200
END process 2184 work : 900
END process 2190 work : 300
end
```

그전 커널들과는 달리 확실하게 연산이 적다고 빨리 끝나지 않는다. (CFS 적용 연산 다르게 줄 때 결과참고)

모두 같은 연산 수

```
juntc@juntc-VirtualBox:~/os/task4/cfs$ ./CFS
Start process 2208 work : 2100
Start process 2209 work : 2100
Start process 2211 work : 2100
Start process 2210 work : 2100
Start process 2212 work : 2100
Start process 2213 work : 2100
END process 2213 work : 2100
Start process 2214 work : 2100
END process 2211 work : 2100
Start process 2215 work : 2100
END process 2209 work : 2100
Start process 2216 work : 2100
END process 2212 work : 2100
Start process 2217 work : 2100
END process 2208 work : 2100

Start process 2218 work : 2100
END process 2210 work : 2100
Start process 2219 work : 2100
END process 2214 work : 2100
Start process 2220 work : 2100
END process 2215 work : 2100
Start process 2221 work : 2100
END process 2217 work : 2100
Start process 2222 work : 2100
END process 2216 work : 2100
Start process 2223 work : 2100
END process 2218 work : 2100

Start process 2224 work : 2100
END process 2219 work : 2100
Start process 2225 work : 2100
END process 2220 work : 2100
Start process 2226 work : 2100
END process 2221 work : 2100
Start process 2227 work : 2100
END process 2222 work : 2100
Start process 2228 work : 2100
END process 2223 work : 2100
END process 2224 work : 2100
END process 2225 work : 2100
END process 2226 work : 2100
END process 2227 work : 2100
END process 2228 work : 2100
end
```

다른 커널의 결과와는 달리 시작 – 종료 가 섞여서 출력된다. (1+2 커널 결과 참고)