아이템 84 프로그램의 동작을 스레드 스케줄러에 기대지 말라

잘 작성된 프로그램은, 스레드 스케줄러에 영향을 받지 않는다.

- 여러 스레드가 실행중일 때, OS의 스레드 스케줄러가 어떤 스레드를 얼마나 오래 실행할지를 정한다.
    - 정상적인 운영체제인 경우 공정하게 처리하지만, 구체적인 스케줄링 정책은 운영체제마다 다를 수 있다.

      → 정확성이나 성능이 스레드 스케줄러에 따라 달라지는 프로그램이면 다른 플랫폼에 이식하기 어렵다.

견고하고 빠르고 이식성 좋은 프로그램을 작성하는 방법

1. **실행 가능한 스레드의 평균적인 수**를 프로세서 수보다 지나치게 많아지지 않도록 하는 것

    > 실행 가능한 스레드 수를 적게 유지하는 주요 기법

    - 각 스레드가 작업을 완료한 후에는 다음 일거리가 생길 때까지 **대기**하도록 하는 것
        - 스레드는 당장 처리해야 할 작업이 없다면 실행돼서는 안 된다.
2. 실행 준비가 된 스레드들은 **맡은 작업을 완료할 때까지 계속 실행**되도록 만드는 것

⇒ 스레드 스케줄러가 고민할 거리가 줄어든다.

피해야 할 상황

1. 스레드는 절대 바쁜 대기 상태가 되면 안 된다. (=공유 객체가 바뀔때까지 쉬지 않고 검사해서는 안된다)
    - 바쁜 대기는 스레드 스케줄러의 변덕에 취약하다
    - 프로세서에 큰 부담을 주어 다른 유용한 작업이 실행될 기회를 박탈한다.

```
public class SlowCountDownLatch {
    private int count;

    public SlowCountDownLatch(int count) {
        if (count < 0)
            throw new IllegalArgumentException(count + " < 0");
        this.count = count;
    }

    public void await() {
        while (true) {
            synchronized (this) {
                if (count == 0)
                    return;
            }
        }
    }

    public synchronized void countDown() {
        if (count != 0)
            count--;
    }
}
```

하나 이상의 스레드가 **필요도 없이 실행 가능한 상태**(바쁜 대기 상태)인 시스템은 흔하게 볼 수 있다. 이런 시스템은 성능과 이식성이 떨어질 수 있다.

▼ CountDownLatch

- 스레드 n개를 실행할 때, **일정 개수의 스레드가 모두 끝날 때까지 기다려야만** 다음으로 진행 할 수 있거나, 다른 스레드를 실행 시킬 수 있는 경우에 사용한다.

2. **Thread.yield**에 의존하면 안된다.
    - 특정 스레드가 다른 스레드들과 비교해 CPU 시간을 충분히 얻지 못해서 간신히 돌아가는 프로그램을 볼때 → Thead.yield를 써서 문제를 고쳐보려는 유혹이 들겠지만, 떨쳐내자!

    ⇒ 증상이 어느정도 호전될 수도 있지만 **이식성은** 그렇지 않다. 처음에는 성능이 좋아졌다가도 두번째에선 효과가 없고 세번째는 오히려 느려지게 될 수 있다 .

    ⇒ Thread.yield는 테스트 할 수단도 없다.

    ⇒ 애플리케이션 구조를 바꿔 **동시에 실행가능한 스레드 수가 적어지도록** 조치하는게 올바른 방법이다.

3. **스레드 우선 순위**에 의존하면 안된다.

- 2번과 같은 상황(특정 스레드의 CPU점유율 높이기)에서 스레드 우선순위를 조절하는 방법도 생각할 수 있지만 역시 좋지 않다.

⇒ 스레드 우선순위는 **자바에서 이식성이 가장 나쁜 특성**에 속한다.

⇒ 스레드 몇개의 우선순위를 조율해서 애플리케이션의 반응 속도를 높이는 것도 타당해보이지만, 정말 그래야 할 상황은 드물고 이식성이 떨어진다.

⇒ 심각한 응답 불가 문제를 스레드 우선순위로 해결하려는 시도는 진짜 원인을 찾아 수정하기 전까지 같은 문제가 반복해서 터져 나올 것이다.

핵심 정리

- 프로그램의 동작을 스레드 스케줄러에 기대지 말자 ⇒ 견고성과 이식성을 모두 해치는 행위다.
- 같은 이유로 Thread.yield 와 스레드 우선순위에 의존해서도 안 된다. 이기능들은 스레드 스케줄러에 제공하는 힌트일 뿐이다. 스레드 우선순위는 이미 잘 동작하는 프로그램의 서비스 품질을 높이기 위해 드물게 쓰일 수는 있지만, 간신히 동작하는 프로그램을 '**고치는 용도'로 사용해서는 절대 안된다.**