

Pintos Project - Alarm clock 의 개선

201810876 문성준

1. 프로젝트 개요 :

Pintos 커널의 timer_sleep() 함수는 현재 busy waiting 방식으로 구현되어 있어 CPU 자원을 비효율적으로 사용하고 있다. busy waiting 방식은 특정 시간이 지날 때까지 CPU를 계속 사용하여 다른 작업을 수행할 수 없게 만드는 문제가 있다. 이를 개선하기 위해 timer_sleep() 함수를 busy waiting 없이 동작하도록 수정하는 프로젝트를 수행했다.

2. 프로젝트 내용 - Alarm clock 의 개선 :

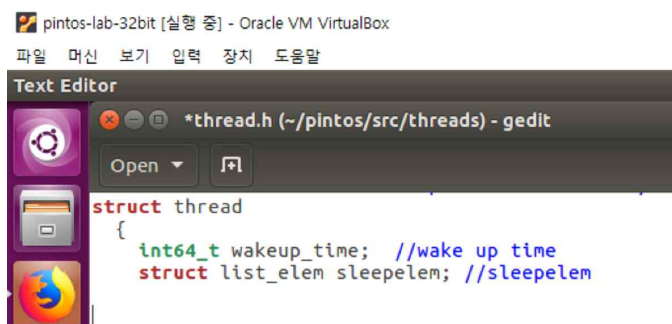
'devices/timer.c'에 정의되어 있는 timer_sleep() 함수를 개선한다. 현재의 timer_sleep()은 기능적으로는 정상 동작하나, busy waiting 방식으로 구현되어 있다. 이를 busy waiting 없이 수행하도록 수정하여 성능을 개선한다.

3. 구현 방법 :

1. 개발 환경 설정

우분투 가상환경을 사용하여 개발을 진행 하였고, 코드 수정을 위해 gedit를 사용하였다. gedit는 GNOME 데스크탑 환경에서 제공하는 텍스트 편집기로, 코드 수정 접근성이 편해진다.

2. 구조체 정의 추가(thread.h 파일)



thread.h 파일에 스레드 구조체 정의를 추가해 스레드가 깨어날 시간을 저장한다. wakeup_time은 스레드가 깨어날 시점을 나타내며, sleepelen은 리스트 요소로 사용된다. 이 구조체를 이용하여 timer_sleep() 함수는 스레드의 깨어날 시점을 추적하고 적절한 시간에 스레드를 깨울 수 있다.

3. 정렬 함수 정의

```
/* sort function (wakeup_time) */
bool wakeup_time_less(const struct list_elem *a, const struct list_elem *b, void *aux UNUSED) {
    const struct thread *t1 = list_entry(a, struct thread, sleep_elem);
    const struct thread *t2 = list_entry(b, struct thread, sleep_elem);
    return t1->wakeup_time < t2->wakeup_time;
}
```

thread가 wakeup_time을 기준으로 스레드를 정렬하여, timer interrupt가 발생할 때마다 다음 thread를 쉽게 찾을 수 있도록 정렬함수를 정의한다. bool 형태로 두 스레드의 wakeup_time을 비교하여 정렬 순서를 결정한다.

4. 타이머 함수 초기화

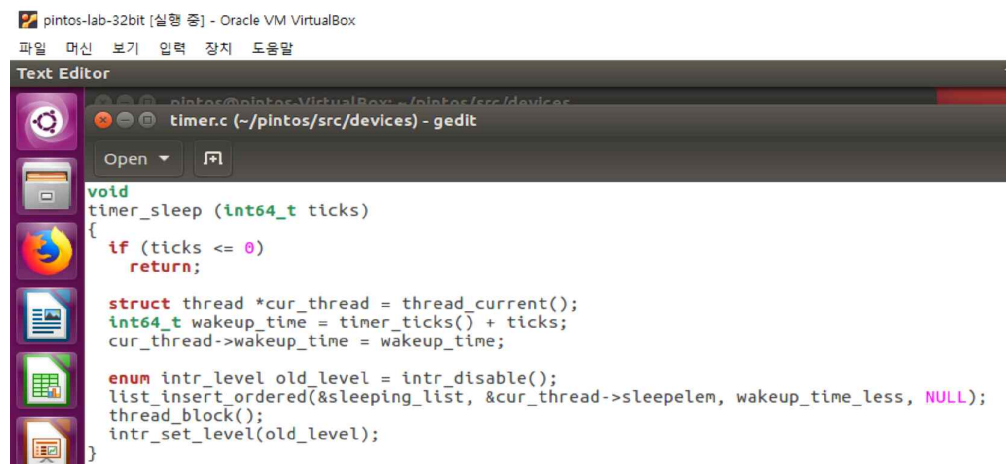
```
static struct list sleeping_list;

.
.
.

void timer_init(void) {
    ...
    list_init(&sleeping_list);
}
```

sleeping_list를 정의하고, timer_init 함수에서 sleeping_list를 초기화 하는 코드를 추가한다.

5. timer_sleep 함수

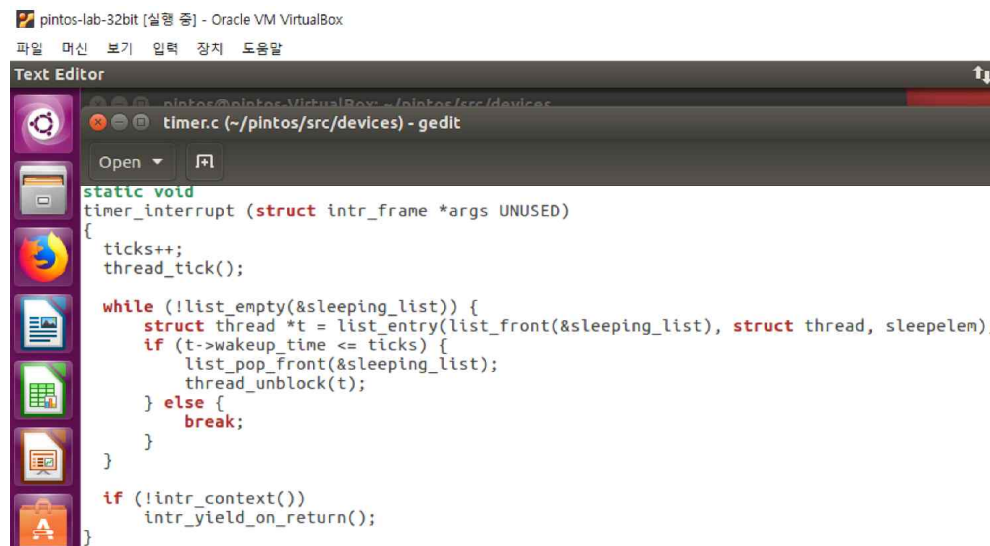


wakeup_time을 설정하고, sleeping_list에 삽입하여, 지정된 시간이 지나면 thread가 깨워질 수 있도록 한다.

코드 설명은 다음과 같다.

- 현재 thread를 주어진 시간(ticks) 동안 sleep으로 만든다.
- ticks가 0 이하인 경우는 sleep할 필요가 없으므로 바로 리턴한다.
- 현재 thread에 현재 시간에 ticks를 더한 값인 wakeup_time을 설정한다.
- sleeping_list에 접근하는 동안 다른 인터럽트가 발생하는 것을 방지하기 위해 인터럽트를 끈다.
- 현재 스레드를 sleeping_list에 wakeup_time을 기준으로 정렬하여 삽입하고, thread를 block 상태로 만든다.
- 인터럽트를 원래 상태로 복원한다.

6. timer_interrupt 함수



ticks가 증가할 때마다 sleeping_list의 thread를 확인하고, 시간이 된 thread를 리스트에서 제거하고 깨운다.

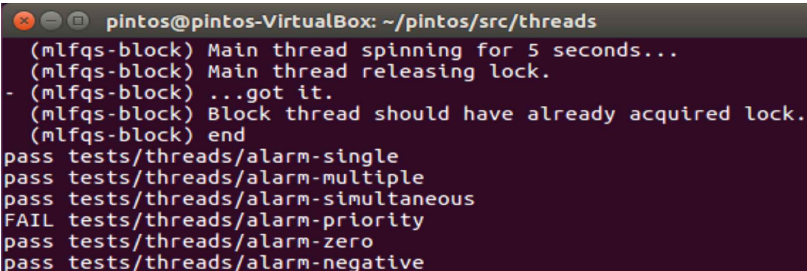
코드 설명은 다음과 같다.

- 현재 thread의 상태를 갱신하기 위해 thread_tick() 함수를 호출한다.
- sleeping_list의 첫 번째 thread를 확인하여 wakeup_time이 현재 ticks 값보다 작거나 같은 경우, 리스트에서 제거하고 해당 스레드를 깨운다.
- 리스트의 첫 번째 thread의 wakeup_time이 현재 ticks 값보다 크다면 반복문을 종료한다.
- 현재 인터럽트가 아닌 경우, intr_yield_on_return() 함수를 호출하여 다른 thread에게 CPU를 양보한다.

4. 테스트

구현된 `timer_sleep()` 함수가 제대로 동작하는지 여부를 다음의 테스트를 통해 확인한다:

- alarm-single
- alarm-multiple
- alarm-simultaneous
- alarm-zero
- alarm-negative



```
pintos@pintos-VirtualBox: ~/pintos/src/threads
(mlfqs-block) Main thread spinning for 5 seconds...
(mlfqs-block) Main thread releasing lock.
- (mlfqs-block) ...got it.
(mlfqs-block) Block thread should have already acquired lock.
(mlfqs-block) end
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
```

alarm-priority 제외한 5개의 테스트 수행 결과가 모두 PASS 되었고, alarm clock 개선이 되었음을 알 수 있다.

5. 프로젝트 결과 (알고리즘 및 자료구조)

timer_sleep()

- `timer_sleep(int64_t ticks)` 호출 시, 현재 시간(`timer_ticks()`)에 ticks를 더해 `wakeup_time`을 설정한다.
- `wakeup_time`을 기준으로 `sleeping_list`에 thread를 정렬하여 삽입하고, block 상태로 만든다.

timer_interrupt()

- `timer_interrupt()` 호출 시 ticks를 증가시킨다.
- `sleeping_list`의 thread의 `wakeup_time`을 확인하여, ticks보다 작거나 같으면 리스트에서 제거하고 unblock한다.

자료구조

`sleeping_list`: 스레드들이 `wakeup_time` 기준으로 정렬된 리스트로, struct list 타입으로 정의된다.

기존의 `timer.c`를 개선 하여 busy waiting을 피하고, 타이머 인터럽트 발생 시 효율적으로 스레드를 관리할 수 있게 되었다. CPU 자원의 효율적인 사용을 가능하게 하여 시스템 전체 성능을 향상시킨다.