

2019320097_조이강

- Priority scheduling을 구현하기 위해서, thread_yield 함수와 thread_unblock 함수의 list_push_back을 list_insert_ordered로 대체합니다.
 - 가장 먼저, list_insert_ordered에 사용할 less 함수 My_Less_Function을 thread.h에 정의합니다.
 - list_insert_ordered는 스레드가 리스트에 우선순위 값에 따라 자신에게 맞는 위치에 삽입될 수 있게 합니다.
 - 우선순위가 높은 스레드는 리스트의 앞에 위치하게 삽입합니다.

```
bool My_Less_Function(struct list_elem *l, struct list_elem *s, void **aux UNUSED);
```

- My_Less_Function은 입력된 두 스레드의 우선순위를 비교합니다.

```
/* Less Function for list_insert_ordered*/
bool My_Less_Function(struct list_elem *l, struct list_elem *s, void **aux UNUSED){
    /*Get thread from list*/
    struct thread *t1 = list_entry(l, struct thread, elem);
    struct thread *t2 = list_entry(s, struct thread, elem);

    /*Return comparing of thread's priority*/
    return (t1->priority) > (t2->priority);
}
```

- list_elem에서 thread 구조체 list_entry를 통해 가져와 t1, t2에 저장합니다.
 - struct thread *t1 = list_entry(l, struct thread, elem);
struct thread *t2 = list_entry(s, struct thread, elem);
 - t1과 t2의 우선순위를 비교하여 반환합니다.
 - return (t1->priority) > (t2->priority);
 - 이를 통해 list_insert_ordered는 t1에 할당된 스레드가 우선순위에 맞게 삽입되게 만듭니다.
- list_push_back을 list_insert_ordered로 교체합니다.

```
void
thread_unblock (struct thread *t)
{
    ...
    /*
    list_push_back (&ready_list, &t->elem);
    */
    list_insert_ordered(&ready_list, &t->elem, &My_Less_Function, NULL);
    ...
}
```

```
void
thread_yield (void)
{
    ...
    /*
    list_push_back (&ready_list, &cur->elem);
    */
    list_insert_ordered(&ready_list, &cur->elem, &My_Less_Function, NULL);
    ...
}
```

- 이를 통해 스레드가 unblock 되거나 yield되어 ready_list에 삽입될 때 자신의 우선순위에 맞는 위치에 삽입되게 됩니다.
- 현재 실행 중인 스레드보다 우선순위가 높은 스레드가 create되거나, set_priority로 우선순위를 높게 할당 받은 경우, Preemption을 진행합니다.
 - 이를 위한 함수 My_Preemption을 정의합니다.

```
/* Preemption Function for thread_create and set_priority*/
void My_Preemption(void);
```

```

/* Preemption Function for thread_create and set_priority*/
void My_Preemption(void) {
    /*Disable interrupt*/
    enum intr_level old_level;
    old_level = intr_disable();

    /*Get current thread and thread of Highest priority*/
    struct thread *cur = thread_current();
    struct list_elem *next = list_begin(&ready_list);

    /*If current thread has lower priority, then call thread_yield */
    if (next != NULL && My_Less_Function(next, &cur->elem, NULL))
        thread_yield();

    intr_set_level(old_level);
}

```

- My_Preemption 함수는 현재 스레드와 ready_list의 첫 번째 스레드의 우선순위를 비교합니다.
 - 새로운 스레드가 create되거나, set_priority 되기 이전에는 현재 스레드의 우선순위는 항상 ready_list의 모든 스레드 보다 높습니다.
 - 만약 새로운 스레드가 현재 스레드보다 높은 우선순위를 가질 경우, ready_list의 맨 앞에 위치하게 됩니다.
 - 따라서 현재 스레드와 ready_list의 맨 앞 스레드를 비교하여 preemption을 구현할 수 있습니다.

```

/*Get current thread and thread of Highest priority*/
struct thread *cur = thread_current();
struct list_elem *next = list_begin(&ready_list);

```

- 현재 스레드를 cur에, ready_list의 가장 앞 원소를 next에 할당합니다.

```

/*If current thread has lower priority, then call thread_yield */
if (next != NULL && My_Less_Function(next, &cur->elem, NULL))
    thread_yield();

```

- ready_list가 비어있지 않으며 next의 우선순위가 더 클 경우 thread_yield를 호출하여 현재 스레드가 cpu를 포기하게 만듭니다.
 - 이때 우선순위 비교를 위해 이전에 정의했던 My_Less_Function을 사용했습니다.
- thread_yield가 호출되면 자동적으로 가장 높은 우선순위의 스레드가 스케줄링되므로, Preemption이 구현됩니다.
- thread_create와 thread_set_priority에 My_Preemption을 추가합니다.

```

tid_t
thread_create (const char *name, int priority,
               thread_func *function, void *aux)
{
    ...
    /* Add to run queue. */
    thread_unblock (t);
    My_Preemption();
    ...
}

```

```

/* Sets the current thread's priority to NEW_PRIORITY. */
void
thread_set_priority (int new_priority)
{
    thread_current ()->priority = new_priority;
    My_Preemption();
}

```

- 이를 통해 스레드가 만들어지거나, 우선순위가 변동될 때 My_Preemption이 호출되어 우선순위에 따른 Preemption이 구현됩니다.

실행 결과

- Priority

```
p2019320097@p2019320097: ~/pintos/src/threads
perl -I../.. ../tests/threads/alarm-simultaneous.ck tests/threads/alarm-simul
taneous tests/threads/alarm-simultaneous.result
pass tests/threads/alarm-simultaneous
pintos -v -k -T 60 --qemu -- -q run alarm-priority < /dev/null 2> tests/thread
s/alarm-priority.errors > tests/threads/alarm-priority.output
perl -I../.. ../tests/threads/alarm-priority.ck tests/threads/alarm-priority
tests/threads/alarm-priority.result
pass tests/threads/alarm-priority
pintos -v -k -T 60 --qemu -- -q run alarm-zero < /dev/null 2> tests/threads/al
arm-zero.errors > tests/threads/alarm-zero.output
perl -I../.. ../tests/threads/alarm-zero.ck tests/threads/alarm-zero tests/th
reads/alarm-zero.result
pass tests/threads/alarm-zero
pintos -v -k -T 60 --qemu -- -q run alarm-negative < /dev/null 2> tests/thread
s/alarm-negative.errors > tests/threads/alarm-negative.output
perl -I../.. ../tests/threads/alarm-negative.ck tests/threads/alarm-negative
tests/threads/alarm-negative.result
pass tests/threads/alarm-negative
pintos -v -k -T 60 --qemu -- -q run priority-change < /dev/null 2> tests/threa
ds/priority-change.errors > tests/threads/priority-change.output
perl -I../.. ../tests/threads/priority-change.ck tests/threads/priority-chang
e tests/threads/priority-change.result
pass tests/threads/priority-change
pintos -v -k -T 60 --qemu -- -q run priority-donate-one < /dev/null 2> tests/t
```

- Preemption

```
p2019320097@p2019320097: ~/pintos/src/threads
21.
(priority-donate-lower) acquire must already have finished.
(priority-donate-lower) Main thread should have priority 21. Actual priority:
21.
(priority-donate-lower) end
pintos -v -k -T 60 --qemu -- -q run priority-fifo < /dev/null 2> tests/threads
/priority-fifo.errors > tests/threads/priority-fifo.output
perl -I../.. ../tests/threads/priority-fifo.ck tests/threads/priority-fifo te
sts/threads/priority-fifo.result
pass tests/threads/priority-fifo
pintos -v -k -T 60 --qemu -- -q run priority-preempt < /dev/null 2> tests/thre
ads/priority-preempt.errors > tests/threads/priority-preempt.output
perl -I../.. ../tests/threads/priority-preempt.ck tests/threads/priority-pree
mpt tests/threads/priority-preempt.result
pass tests/threads/priority-preempt
pintos -v -k -T 60 --qemu -- -q run priority-sema < /dev/null 2> tests/threads
/priority-sema.errors > tests/threads/priority-sema.output
perl -I../.. ../tests/threads/priority-sema.ck tests/threads/priority-sema te
sts/threads/priority-sema.result
FAIL tests/threads/priority-sema
Test output failed to match any acceptable form.

Acceptable output:
(priority-sema) begin
```