Lab 2: Alarm Clock in Pintos

Group 43

Shivneshwar Velayutham and Madhumitha Venkatesan

Data structures

Two new fields are added to the thread class.

1. **is_sleeping** is a flag that tells us if a thread is sleeping or not.
2. **wakeup_time** is the time when the thread has to wake up from sleep.

Two new methods have been added to thread class.

void thread_start_sleeping(int64_t sleep_start, int64_t sleep_time)

Make a thread to go to sleep from sleep_start with sleep time = sleep_time

void thread_try_wakeup(struct thread *t, int64_t *current_time)

See if the thread is ready to wakeup from sleeping

Algorithms

When **timer_sleep (int64_t ticks)** is called then **void thread_start_sleeping(int64_t sleep_start, int64_t sleep_time)** is called which sets the thread to sleep and it's waking up time before calling **thread_block()**. Here the thread goes into BLOCKED state.

The waking up logic is written in **timer_interrupt ()** where each thread is checked if it's ready to wake up using **void thread_try_wakeup(struct thread *t, int64_t *current_time)**. If the thread is sleeping and it's waking up time is equal to or past the current time then we try to wake up the thread by calling **thread_unblock()**. Here the thread goes into READY state.

Synchronization

Whenever there is a state change in **is_sleeping** it becomes a critical section because the thread might modify above flag without actually blocking/unblocking the thread. Hence whenever there is a state change then interrupts are disabled until the critical section is over.

Rationale

We went with **wakeup_time** which signifies the actual wake up time instead of maintaining the amount of sleep left for the thread since that would mean that for every interrupt we must decrement the sleep time left. In our solutions the number of operations is reduced by comparing the current time with the **wakeup_time**.