Design Document of Pintos
Project 1: Threads

Tianyi Zhang
School of Information Science and Technology
20185332??
zhangty2@shanghaitech.edu.cn

Haoran Dang
School of Information Science and Technology
2018533259
danghr@shanghaitech.edu.cn

PRELIMINARIES

Acknowledgements

- https://www.cnblogs.com/laiy/p/pintos_project1_thread.html: We read the passage to get familiar with current code structure and how it works.
- https://www.runoob.com/cprogramming/c-enum.html: We read it to understand `enum` in the code.

1 ALARM CLOCK

1.1 Data Structures

1.1.1 Copy here the declaration of each new or changed 'struct' or 'struct' member, global or static variable, 'typedef', or enumeration. Identify the purpose of each in 25 words or less.

- `int64_t sleeping_ticks` in `struct thread`: a counter of remaining sleeping ticks.

1.2 Algorithms

1.2.1 Briefly describe what happens in a call to `timer_sleep()`*, including the effects of the timer interrupt handler.* When `timer_sleep` is invoked, it set a counter inside the thread as the countdown of remaining sleeping ticks, and calls `thread_block` to avoid it from running.

Then in timer interrupt (which should be called in each tick), we check this status of all threads by using `thread_foreach`. In the counter of these threads, `0` is for not sleeping and positive number stands for the remaining ticks.

We just simply skip the threads with counter value `-1`. When we find that the counter of a thread reaches `0`, we unblock the thread with `thread_unblock`, which will unblock it and put it into the ready list.

1.2.2 What steps are taken to minimize the amount of time spent in the timer interrupt handler?

1.3 Synchronization

1.3.1 How are race conditions avoided when multiple threads call `timer_sleep()` *simultaneously?*

1.3.2 How are race conditions avoided when a timer interrupt occurs during a call to `timer_sleep()`*?* Inspired by function `timer_ticks`, we can use

```
enum intr_level old_level = intr_disable ();
...
intr_set_level (old_level);
```

to ensure an atomic operation. First we call `intr_disable`, which will make the process uninterruptible and returns the old status. Then we do our operations, and since the process cannot be interrupted, the operations are atomic. Finally, we restore the interrupt status by `intr_set_level`.

1.4 Rationale

1.4.1 Why did you choose this design? In what ways is it superior to another design you considered? `thread_foreach`, `thread_block` and `thread_unblock` is mentioned in the project guide.