COMP312005 Operating System, Spring 2019

Project 1 : Alarm Clock

**---- GROUP ----**

Hyosang Park <latter2005@gmail.com

Yongho Lee <jhyn207@gmail.com>

Jisu Kim <wltn4511@gmail.com>

**---- PRELIMINARIES ----**

Contribution : 1:1:1

Collaboration : debugging, implementation of code, idea sharing.

Hyosang Park : report editing, code editing.

Yongho Lee : researching information from pintos maunal, report editing.

Jisu Kim : report writing.

**ALARM CLOCK**

===========

---- DATA STRUCTURES ----

>> A1: 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.

**Thread.h**

-in struct thread,

Int64\_t*sleep tick* is added, to store tick to wake thread.

**Thread.c**

-static struct list *sleep\_list*

List of sleeping threads.

-static int64\_t *smallest\_tick*

Variable smallest tick is set as smallest tick of thread in the list.

---- ALGORITHMS ----

>> A2: Briefly describe what happens in a call to timer\_sleep(),

>> including the effects of the timer interrupt handler.

>> A3: What steps are taken to minimize the amount of time spent in

>> the timer interrupt handler?

**<Added functions>**

**Thread.c**

-void **set\_smallest\_tick**(*int64\_t tmp*)

Function to set *smallest tick*.

-int64\_t **get\_smallest\_tick**()

Function to return *smallest\_tick*.

-void **thread\_sleep**(*int64\_t ticks*)

Gets target thread by **thread\_current**(), puts target thread in *sleep list*. Thread is inserted to appropriate position in list by comparing ticks of each thread in the list. Then, blocks target thread and blocks interrupt by **intr\_disable**()

-void **thread\_awake**(*int64\_t ticks*)

Checks sleep list, and wakes threads(remove from sleep list) with ticks smaller than current tick. After, sets *smallest tick* to smallest tick of thread in *sleep list*.

**Timer.c**

-void **timer\_interrupt**(*struct intr frame \*args UNUSED*)

Edited to check current tick and thread’s tick

by **get\_smallest\_tick**(). Wakes up if current tick is same or less than get smallest tick.

-void **timer\_sleep**(*int\_64 ticks*)

Edited to call **thread\_sleep**(*start+ticks*)

A2.

When **timer\_sleep()** is invoked, it invokes **thread\_sleep**(start+ticks). start is current\_tick, ticks is amount of tick thread is going to sleep.

In **thread\_sleep**, target thread is inserted into *sleep*\_*list*. After inserted to the list, it blocks target thread and signal to the blocked thread.

**timer\_interrupt()** periodically increases tick. when it’s time to wake thread in *sleep*\_*list*(when tick >= smallest\_tick), **timer\_interrupt()** invokes **thread \_awake()**

. **thread\_awake()** checks *sleep*\_*list*, and wake threads with *sleep*\_*tick* smaller than current tick. it unblocks thread, and removes thread from sleep\_list.

new *smallest*\_*tick* is set as smallest sleep tick in the *sleep*\_*list*.

A3.

To minimize the amount of time spent in the timer interrupt handler, interrupt handler only increases tick and checks whether it’s time to wake thread in sleep\_list or not. By using global variable *smallest*\_*tick*, interrupt handler can easily check the time to wake thread.

Detailed functions are implemented in **thread\_sleep()** and **thread\_awake().**

By sorting threads by time-order in *sleep list* at **thread\_sleep()**, **thread\_ awake()** can easily find which thread should be awaken.