Università
della
Svizzera
italiana

Faculty
of
Informatics

Operating Systems

PintOS: Report Project 1 - Timer Sleep

Group 11
Machado Ferreira Simone, Zeneli Jora, Del Pozzo Daniele

Spring Semester, March 13, 2025

1 Questions

Answer the following questions:

1. How do you keep track of sleeping threads?
2. How do you avoid busy waiting in your implementation?
3. How do you wake up sleeping threads?

List here the files you created/modified:

- **timer.c** (modified)
- **thread.c** (modified)
- **thread.h** (modified)

2 Answers

1. Through a LinkedList named 'threads_asleep' that contains all sleeping threads. The list is sorted in increasing order based on wakeup times to make the selection of which threads to wake up more efficient.
2. Instead of having a while loop that checks whether enough time has elapsed, we instead just store the time that the thread should be woken up in a variable called 'wakeup' inside thread and then we just block it, insert it in the 'threads_asleep list and from there the thread is forgotten about until it's time to wake it up.
3. This is handled by the timer_interrupt handler. It works by going over the list of threads that are asleep at the moment, and wakes up all threads whose sleep timer 'wakeup' has expired. As soon as it finds a thread whose timer hasn't expired it stops checking since they're sorted, thus not wasting time checking threads that are still in sleep mode.

3 Further comments

1. **timer.c** changes:

   changed the timer_sleep function to avoid busy waiting by deleting the while loop that yields the thread, instead now a new function called ′thread_sleep′ is called that handles the thread sleeping process in case the thread wakeup timer hasn′t already expired.

   Also expanded the timer_interrupt handler. Now the function accesses the ′threads_asleep′ list and, as stated in answer 3, wakes up all eligible threads and removes them from the asleep list.

2. **thread.c** changes:

   We initialize the ′threads_asleep′ list. Also we created a new function named ′thread_sleep′ that handles the process to put threads in the sleeping list. This is done by changing the thread status to ′THREAD_BLOCKED′ and the ′wakeup′ parameter value to the time the thread has to be woken up, then the thread is blocked and put in the ′threads_asleep′ list.

   To sort the sleeping threads in the asleep list we also created an auxiliary function that takes two threads and simply returns the comparison between wakeup times between two threads and we fed it to the ′list_insert_ordered′ function

3. **thread.h** changes:

   created the ′threads_asleep′ linked list to house all sleeping threads and added two new fields to the thread struct: a list element for sleeping threads list and the wakeup that is used in case the thread is asleep.