Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

Operating Systems **Spring - 2022**

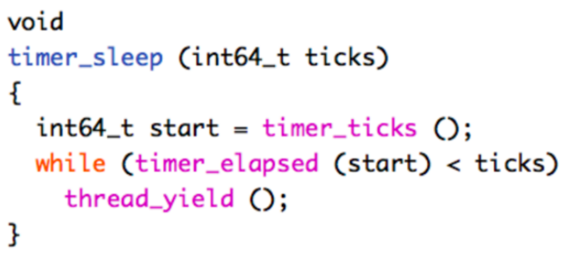
Prof. Fernando Pedone, Eliã Rafael L Batista, Nenad Milosevic

Project 2 - Timer Sleep PintOS

So far, we have discussed how to build simple tests in PintOS and how to work with the list data structure included in the basic C Library present in PintOS. In this second (**graded**) project, your task is to re-implement the threads ***"sleep"*** functionality.

The thread's sleep function

The current thread sleep functionality is implemented in the ***"timer_sleep()"*** function located in the ***"devices/timer.c"*** file in PintOS. This function is executed when user processes call the sleep system-call. The current implementation is like so:



```
void
timer_sleep (int64_t ticks)
{
  int64_t start = timer_ticks ();
  while (timer_elapsed (start) < ticks)
    thread_yield ();
}
```

As you may notice, the problem here is that this implementation uses a BUSY WAIT approach and your task is to mitigate that.

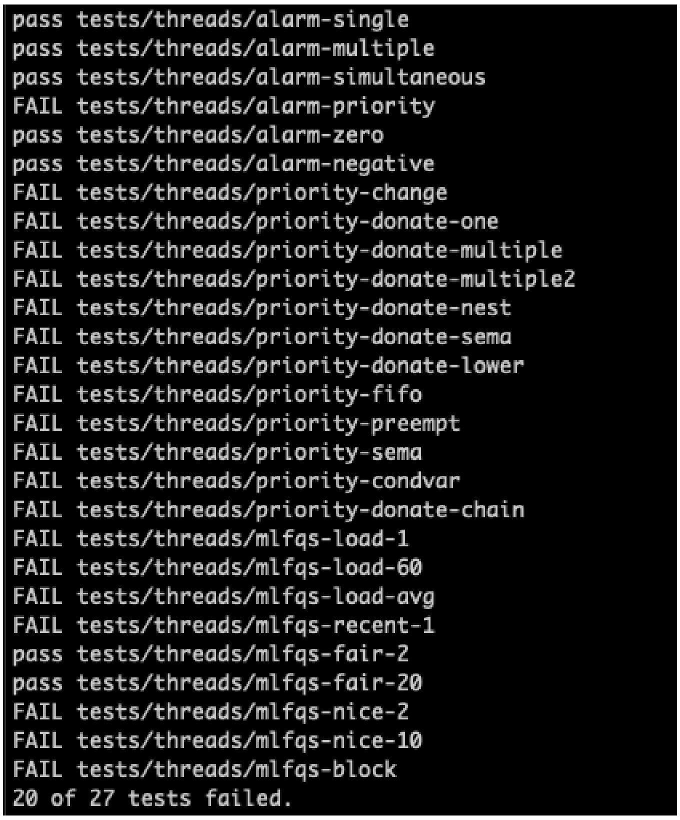
Re-implement this function to:

- Avoid busy waiting
- Not brake tests that already pass!
  - to see the tests that are being executed and already passing run `make check` in the threads directory before starting your implementation
- Ideally, be efficient

Hints

- Use ***"thread_block()"*** and keep track of sleeping threads (using a list, for example)
  - ***"thread_block()"*** is a function defined in *threads/thread.c* file and puts the current thread to sleep. It will not be scheduled again until awoken by ***"thread_unblock()"*** function.
- During timer interrupts, ***"thread_unblock()"*** sleeping threads if their timer expired
- Disable interrupts to protect critical sections

A correct implementation would not make any of the current tests start failling, and an execution of `make check` should print the following result:



```
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
FAIL tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiple2
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-sema
FAIL tests/threads/priority-donate-lower
FAIL tests/threads/priority-fifo
FAIL tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
FAIL tests/threads/mlfqs-load-1
FAIL tests/threads/mlfqs-load-60
FAIL tests/threads/mlfqs-load-avg
FAIL tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
FAIL tests/threads/mlfqs-nice-2
FAIL tests/threads/mlfqs-nice-10
FAIL tests/threads/mlfqs-block
20 of 27 tests failed.
```

Readings

- PintOS documentation
  - Chapter 2 up to 2.2.2
  - Appendix A.2
  - Appendix A.3.1
  - Skim through Appendix A.1