CMPS111 Spring 2018 : Lab 1

In this lab you will make minor modifications and minor additions to the source code for the teaching operating system Pintos.

Specifically, you will implement a more efficient version of the Pintos system call `timer_sleep` found in `src/devices/timer.c` in the Pintos distribution supplied for this lab.

This lab is worth 5% of your final grade.

Submissions are due NO LATER than 23:59, Wednesday April 11, 2018 (one week)

Setup

SSH in to one of the two CMPS111 teaching servers using your CruzID Blue password:

```
$ ssh <cruzid>@noggin.soe.ucsc.edu   ( use Putty http://www.putty.org/ if on Windows )
Or  $ ssh <cruzid>@nogbad.soe.ucsc.edu
```

Authenticate with Kerberos:

```
$ kinit <cruzid>@CATS.UCSC.EDU
```

Authenticate with AFS:

```
$ aklog
```

Create a suitable place to work: ***(only do this the first time you log in)***

```
$ mkdir -p CMPS111/Lab1
$ cd CMPS111/Lab1
```

Install the lab environment: ***(only do this once)***

```
$ tar xvf /var/classes/CMPS111/Spring18/Lab1.tar.gz
```

Build Pintos:

```
$ cd ~/CMPS111/Lab1/pintos/src/threads      ( always work in this directory )
$ make
```

Also try:

```
$ make check        ( runs the required functional tests - see below )
$ make grade        ( tells you what grade you will get - see below )
```

Additional Information

`void timer_sleep(int64_t ticks)` should suspend execution of the calling thread until time has advanced by at least the requested number of ticks. Unless the system is otherwise idle, the thread need not wake up after exactly ticks. It is entirely valid to just put it on the ready queue after they have waited for the right number of ticks.

The `ticks` argument is expressed in timer ticks, not in milliseconds or any other unit. Do not change this data type unless you want lots of the tests to fail.

Separate functions `timer_msleep`, `timer_usleep`, and `timer_nsleep` exist for sleeping a specific number of milliseconds, microseconds, and nanoseconds respectively, but these will call `timer_sleep` as appropriate. You do not need to modify them.

Accessing the teaching servers file systems from your personal computer

You home directories on the teaching servers are UCSC AFS, i.e. they are the same as if you logged into the UCSC Unix Timeshare. To access files in your home directory to edit them using an IDE on your laptop, you can do one of two things:

- Use an SSH synchronization client like Cyberduck: https://cyberduck.io/
  - Configure Cyberduck to point at either of the teaching servers
  - Connect with your CruzID and Blue Password

- Install an AFS client on your laptop: https://www.auristor.com/openafs/client-installer/
  - The Kerberos realm is: CATS.UCSC.EDU
  - The OpenAFS Cell is: cats.ucsc.edu
  - Use your CruzID and Blue Password
  - This is a far from trivial setup. If you get stuck, switch to using something like Cyberduck.
  - **DO NOT CONTACT UCSC AFS SUPPORT FOR ANY REASON WHATSOEVER**

Requirements

Basic:

- You have modified the implementation of `void timer_sleep(int64_t ticks)` function found in `src/devices/timer.c`

- Your modified implementation passes the following functional tests:
  - alarm-single
  - alarm-multiple
  - alarm-simultaneous
  - alarm-zero
  - alarm-negative

- Your implementation is demonstrably efficient (a non-functional test).

What steps should I take to tackle this?

Come to the sections and ask.

How much code will I need to write?

A model solution that satisfies all requirements adds approximately 20 lines of executable code.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

   a. Tests pass [5% per test] (25%)
   b. Your implementation is demonstrably more efficient than the original (75%)

2. (-100%) **Did you give credit where credit is due?**

   a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%)

   b. Your submission is determined to be a copy of another past or current CMPS111 student's submission (-100%)

What to submit

In a command prompt:

```
$ cd ~/CMPS111/Lab1/pintos/src/threads
$ make submit
```

This creates a gzipped tar archive named `CMPS111-Lab1.tar.gz` in your home directory.

UPLOAD THIS FILE TO THE APPROPRIATE CANVAS ASSIGNMENT.

Looking ahead

Subsequent labs will require a short report and will have additional assessment criteria, including:

Is it well written?

Marks awarded for:

- Compilation free of errors and/or warnings.
- Clarity
- Modularity

Marks deducted for:

- Failing to do any of the above
- Not following C language good practice (e.g. don't use magic numbers)

I strongly recommend you consider these issues in this lab to get some practice in.

§

