**Operating Systems Laboratory (CS39002)**

**Spring Semester 2020-2021**

**Assignment 3:** Introduction to PintOS and improving threads in PintOS

**Assignment given on**: January 28, 2021

**Assignment deadline**: February 11, 2021, 1:00 PM

PintOS (Ben Pfaff et al.) [http://pintos-os.org/SIGCSE2009-Pintos.pdf] is a popular minimal operating used at many Universities – in India and overseas – to train students in the internal details of the Unix-like operating systems. PintOS is minimal since many OS capabilities are missing and you will be asked to implement them by modifying the codebase.

The primary document that we use in this subject was originally written at the Stanford University [https://web.stanford.edu/class/cs140/projects/pintos/pintos.pdf]. The more detailed version of PintOS and the projects used in Stanford is here: https://web.stanford.edu/class/cs140/projects/pintos/pintos.html (This should suffice to give you a basic overview of PintOS).

This warming-up assignment has two parts: In the first one we ask you to install pintOS on a virtual machine with all the dependencies. In the second one we ask you to improve the thread implementation supported by PintOS.

## Part-1: Installation of PintOS (20 marks)

We can install PintOS in many operating systems (like Windows, linux, mac) with slightly different set-up instructions for each platform. However, to ensure the uniformity of implementation (as well as not limit our ability to help you) we chose to use virtual boxes for our PintOS assignments.

For installing virtual box you need two things: (i) an operating system image and (ii) a software like oracle virtual box which can load the image and create virtual operating system (technically a Type-2 hypervisor).

- Download Oracle virtual box from here: https://www.virtualbox.org/wiki/Downloads
- Download the image from here: https://www.osboxes.org/ubuntu/#ubuntu-14_04-vbox (We will use the “Ubuntu 14.04.6 Trusty Tahr” version)

Open the image using virtualbox, you will be asked to create a virtual machine, we suggest to assign it more than 2 GB or memory and more than 4 GB of physical storage. You need to run PintOS in this virtual machine. Within your virtual machine PintOS will run in a simulator, we will use QEMU for this purpose. In your virtual box follow the following steps:

Installing PintOS with QEMU:

1. Install qemu
```
> sudo apt-get install qemu
```

2. Add a softlink
```
> sudo ln -s /usr/bin/qemu-system-x86_64 /usr/bin/qemu
```

3. Download PintOS
   http://www.stanford.edu/class/cs140/projects/pintos/pintos.tar.gz
   Extract PintOS to home folder (say $HOME)

4. Setup GDBMACROS
   Open the script 'pintos-gdb' (in $HOME/pintos/src/utils. Find the variable GDBMACROS and set it to point to '$HOME/pintos/src/misc/gdb-macros'.

5. Compile utilities
   ```
   > cd $HOME/pintos/src/utils
   ```
   Edit "Makefile" in the current directory and replace "LDFLAGS = -lm" with "LDLIBS = -lm"
   ```
   > make
   ```

6. Adjust Make.vars
   Open the file "$HOME/pintos/src/threads/Make.vars" and change the last line to: SIMULATOR = –qemu

7. Compile PintOS Kernel
   ```
   > cd $HOME/pintos/src/threads/
   > make
   ```

8. Make these three changes in "$HOME/pintos/src/utils/pintos"
   a) Change line no. 103 to "$sim = "qemu" if !defined $sim;" to enforce QEMU as simulator
   b) On line no. 259, replace "kernel.bin" to path pointing to kernel.bin file at "$HOME/pintos/src/threads/build/kernel.bin".
   c) Comment out all the lines in subroutine SIGVTALRM(lines 927 to 935)
9. Open "$HOME/pintos/src/utils/Pintos.pm" and replace "loader.bin" at line no. 362 to point to loader.bin located at "$HOME/pintos/src/threads/build/loader.bin"

10. Update PATH
    Add below mentioned line at the end of $HOME/.bashrc file:
    ```
    export PATH=$HOME/pintos/src/utils:$PATH
    ```
    Restart the terminal or execute
    ```
    > source $HOME/.bashrc
    ```

11. Finally, it's time to check if you have done everything ok: Run pintOS
    ```
    > cd $HOME/pintos/src/utils/
    ```
    Verify installation of pintos using following command.
    ```
    > pintos run alarm-multiple
    ```
    This should create 5 threads and sleep for some predefined times.

**Deliverables:**

You need to show your assigned TA that you installed pintos correctly and the "pintos run alarm-multiple" is working as expected.

## Part 2: Implement an Alarm clock (80 marks)

**Problem description:**

By default, PintOS kernel provides busy waiting when a thread needs to block. You need to change the behavior and block the thread till a fixed amount of timer ticks pass. You will work primarily in the "threads" directory of the PintOS distribution.

Before you start coding, go through this link: https://web.stanford.edu/class/cs140/projects/pintos/pintos_2.html#SEC18 . it will provide you an understanding of the gist and functionalities of each relevant file.

**Task:**

Reimplement timer_sleep(), defined in devices/timer.c. Currently the implementation "busy waits" that is it spins in a loop checking the current time and calling thread_yield() until enough time has gone by. Here is the description of the function.

Function: void **timer_sleep** (int64_t ticks)
Suspends execution of the calling thread until time has advanced by at least x timer ticks. Unless the system is otherwise idle, the thread need not wake up after exactly x ticks. Just put it on the ready queue after they have waited for the right amount of time.

Reimplement the function to avoid busy waiting.

**Submission Guideline:**

You need to upload a zip containing the files you changed along with a design document in Moodle. There should be one submission from each group. Name your zip file as "**Assgn3_<groupno>.zip**". The zip file should contain:

- The files that you changed.
- A design document. You can find the template for design document here: https://web.stanford.edu/class/cs140/projects/pintos/threads.tmpl. Fill it up according to your implementation and include it in your zip file.

**Grading scheme for part 2:**

The total marks for this part of the assignment (80 marks) is divided as follows.

| | |
|---|---|
| Demo the implementation to your assigned TAs and show that your code works as intended | **30 marks** |
| During demo you need to demonstrate that you have indeed implemented the algorithm promised in the design document | **50 marks** |