Assignment 4
Operating Systems (PG)

Part A: Reading assignment
Objective:

To understand the internals of PINTOS
Pintos is an instructional operating system, complete with documentation and readymade, modular projects that introduce students to the principles of multiprogramming, scheduling, virtual memory, and filesystems. By allowing students to run their work product on actual hardware, while simultaneously benefiting from debugging and dynamic analysis tools provided in simulated and emulated environments, Pintos increases student engagement. Unlike tailored versions of commercial or open source OS such as Linux, Pintos is designed from the ground up from an educational perspective. It has been used by multiple institutions for a number of years and is available for wider use.

Understanding PINTOS basics from reading material
Building PINTOS executable from source code and run it on virtual machine
Understanding basics of utilities like GNU MAKE,etc
Get familiar with source code – files and data structures.
Link :
Introduction :http://www.stanford.edu/class/cs140/projects/pintos/pintos_1.html#SEC1
Source code: http://www.stanford.edu/class/cs140/projects/pintos/pintos.tar.gz

Part B:
Objective:
To understand the initial thread system provided by Pintos kernel and implement thread blocking mechanism.

Details:

Pintos already provides following functionalities:
1. Thread creation and completion,
2. A very basic scheduler,
3. Synchronization primitives.
Understand the thread system provided by Pintos kernel using following link under section A.2:

Pintos Thread Implementation

Also, go through following link that provides the functionalities of files under src/threads:

Pintos Thread Directory Demystified

To implement:

Reimplement timer_sleep(), defined in devices/timer.c. Although a working implementation is provided, it "busy waits," that is, it spins in a loop checking the current time and calling thread_yield() until enough time has gone by. Reimplement it to avoid busy waiting.

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. timer_sleep() is useful for threads that operate in real-time, e.g. for blinking the cursor once per second.

The argument to timer_sleep() is expressed in timer ticks, not in milliseconds or any another unit. There are TIMER_FREQ timer ticks per second, where TIMER_FREQ is a macro defined in devices/timer.h. The default value is 100. We don't recommend changing this value, because any change is likely to cause many of the tests to fail. Separate functions timer_msleep(), timer_usleep(), and timer_nsleep() do exist for sleeping a specific number of milliseconds, microseconds, or nanoseconds, respectively, but these will call timer_sleep() automatically when necessary. You do not need to modify them.

Deadline: 1st October, Tuesday, 11:59PM
Upload Format: .tar.gz
Create a folder named your roll number.
Copy the /src directory from PINTOS directory to this folder.
Create a tar.gz named “Assignment4.tar.gz” and upload it.