Wissam Razouki

CSE 4300

Programming Assignment 4

**Part A**

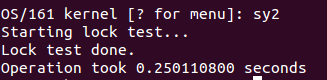
“synch.h” changes:

* Added a thread structure called “holder” within the lock structure to keep track of the thread that currently holds the lock.

“synch.c” changes:

* “lock\_create()” – Initialized the holding thread “holder” to “NULL”.
* “lock\_destroy()” – No changes.
* “lock\_acquire()” – The function begins by asserting the lock is not “NULL”. Then, the priority is set so that all other interrupts are disabled and the operation can be done atomically. The current thread is put to sleep until the thread that holds the lock releases it. To check for this, we run a while loop until “holder” is “NULL”. Once it is, the “holder” for the lock is set to “curthread”. Finally, the interrupt for this process is enabled.
* “lock\_release()” – Similarly to “lock\_acquire()”, we begin by making the same assertion and disabling interrupts. The lock is released by setting the holding thread “holder” to “NULL”. The threads on the lock are woken up, and the interrupt is enabled.
* “lock\_do\_i\_hold()” – Again, we begin by making the same assertion and disabling interrupts. We check if the current thread holds the lock by checking if “holder” is equal to “curthread”. If so, the return value is 1. Otherwise, it is 0. The interrupt for this process is enabled, and the return value is returned.

**Output:**



From the image above, we can see that running the “sy2” command within OS161 initializes the lock test. The test is completed successfully.