University of Massachusetts Dartmouth
CIS 370, Fall 2013
11/05/2013, 11/07/2013
Lab 8 - Thread Synchronization with Semaphores in UNIX
Due: 11/12/2013 (Tuesday), 11/14/2013 (Thursday)

Objective
In this lab, you are going to understand and experiment with Linux Semaphores to provide mutual exclusion between Threads trying to access a critical section of code.

Description
Download the template code and modify it to include various Semaphore operations which will ensure that the threads will always increment the `critical_value` variable to 2,000,000.

In order to complete the above task, you will first have to implement the Semaphore `wait()` and `signal()` functions using the UNIX Semaphore procedures described in the slides. Once you've done this, you can call these procedures to provide mutual exclusion between the Threads in the template code.

Don't forget to remove the Semaphore from the system once it's no longer needed!

*NOTE: You should use only the Semaphore functionality provided in `sys/sem.h` and NOT `semaphore.h` to complete this lab. Solutions using `semaphore.h` will receive no credit!

The template code may be found at:
www.cis.umassd.edu/~jplante/cis370/lab08/lab8_template.c

The Semaphore header and implementation files may be found at:
www.cis.umassd.edu/~jplante/cis370/lab08/pv.h
www.cis.umassd.edu/~jplante/cis370/lab08/pv_template.c

*Be sure to include your last name in the names of your submitted files (I'd prefer not to have 50 files named `pv_template.c`). You do not need to submit `pv.h` since you don't need to make any modifications to that file.

Resources
- **Chapter 8**, *Unix System Programming*, Haviland and Gray, Prentice Hall (Lab Book)
- **Chapter 4**, *Operating System Concepts*, Silberschatz, Galvin, Gagne (Course Text Book)
- **Chapter 6**, *Operating System Concepts*, Silberschatz, Galvin, Gagne (Course Text Book)
- http://www.advancedlinuxprogramming.com/alp-folder/alp-ch04-threads.pdf
- http://www.advancedlinuxprogramming.com/alp-folder/alp-ch05-ipc.pdf