CMSI 387-01

OPERATING SYSTEMS

Spring 2013

Assignment 0411 Feedback

Britain Southwick

All of Time and Space

3b — You did a great job with tracking down your selected papers and recording their connections and differences. Your work is most notable in terms of its specificity of detail, and you did a particularly good job with summarizing the credentials of each author. Well done. (+)

4d — Excellent work with finding papers from the digital library. (+)

4e — Your commits and messages were quite appropriate to the work performed. (+)

4f — One entry submitted on time, with the rest coming in around a half-day later. (|)

The Dining Philosophers Problem

2d — The core algorithm for dining philosophers is in there, but you got tripped up by the API. I guess the analogous code from *bounded-buffer* was not sufficient here. I'll go easy on you for this outcome because the core semaphore calls are generally right, and your issues center mainly on just figuring out how to use the API correctly. (|)

4a — Now, for *this* outcome, we can focus on functional issues. Outside of the whole semaphore API business, your code needs to use its constants more consistently; in one case, this misuse actually would cause a spurious critical section violation message. Your use of philosopher names is entertaining, but this makes your program harder to "eyeball" for correctness because the user would have to memorize which philosopher is in which position. Your "play-by-play"-style output gives some hint of what is going on, but ideally, a "state of the table" display showing which forks are held or not at a single glance would be ideal for truly seeing what is going on. Add all of this up and you get a program that is headed in the right direction but has a good number of issues to resolve. (/)

4b — Your largely maintained the separation of concerns from the *bounded-buffer* example, but missed some definitions that should have gone in a header file and did not rename *producer.c* to something more appropriate. Your philosopher names are also unnecessarily duplicated. (|)

4c — Your code is decently readable; if anything, your code in *philo-cs.c* could use better spacing and possibly more comments, to guide you through what is happening in each phase of getting or putting forks. (|)

4d — You did a decent job with figuring out the core structure of the code, but less so with learning the semaphore API. (|)

4e — Your commit frequency is commensurate to the work done, and your commit messages give some insight into your development process. (+)

4f — Submitted on time, though buggy. But still on time. (+)