CMPS 455

Andrew Shullaw
c00161818

October 8, 2021

Project 2

1 As part of your report, answer the following questions about Task 1:

1. *When implementing and testing your solution, did you notice any deadlock? Deadlock occurs when threads cannot progress due to improperly controlled access to critical resources. (In this case, the critical resources are the chopsticks.) How did you solve any deadlock problems?*

1. I did not experience deadlock, but I did experience an infinite waiting loop where I had the philosophers waiting until everyone was standing to leave. I changed the algorithm to allow philosophers to leave once the meal goal was met.

2. *Make the philosophers yield between attempting to pick up the left and right chopsticks. Run at least 3 tests with various parameters and -rs seeds. Record your findings and explain your results. Undo any changes made to accommodate this question before submitting your assignment. bad input to cause an error? Consider situations other than typing input when prompted.*

2. I thought I was going crazy, but then I remembered that my philosophers cannot pick up a stick without both being available. It had no difference no matter what seed or parameters I attempted.

2

As part of your report on this project, answer the following questions about Task 2:

1. *Run your solutions to Task 1 and Task 2 with the same parameters, including -rs seeds. Note the number of ticks that NachOS runs. Do this for at least 3 different sets of parameters. Record your results in a table, including -rs seeds, number of philosophers, and number of meals. Explain your findings.*

1.

-rs 10 P:2 M:1	-rs 10 P:2 M:1	-rs 20 P:5 M:5	-rs 20 P:5 M:5	-rs 30 P:100 M:100	-rs 30 P:100 M:100
Ticks	Ticks	Ticks	Ticks	Ticks	Ticks
total 278	total 378	total 1394	total 278	total 40284	total 61984
idle 8	idle 108	idle 14	idle 8	idle 134	idle 14
system 270	system 270	system 1380	system 270	system 40150	system 61970

The CPU spends more time idle with the semaphore for small instances, but for large instances (100,100) it outperforms the busy waiting loop. The total CPU time for the semaphore is much higher as well. There are a lot of print statements, so that may be skewing things, but I suppose you could look at that like an I/O bottleneck or just the fact that it is *processing* more with the semaphore, therefore having more ticks.

2. *Make the philosophers yield between attempting to pick up the left and right chopsticks. Run at least 3 tests with various parameters and -rs seeds. Record your findings and explain your results. Undo any changes made to accommodate this question before submitting your assignment.*

2. Yielding between signals causes the philosophers to appear as if they are taking turns picking up sticks in rotation, rather than picking both sticks up at the same time. It is much easier to read the output when this happens for sure. Very cool!

3

As part of your report, answer the following question about Task 3:

1. Did you experience any deadlock when testing this task? How was it different from Task 2?

1. Yes, I wasn't deleting the mail that I read. This was easier than Task 2 with respect to synchronization, but as far as pointers are 3d arrays... *mais las.* I took this convoluted path of passing an object to the Fork as an (int) and it just wasted my life away for a few days. The most difficult part of this task, along with `C` in general, is pointers.

4

1. I dont understand it 2. no idea i never tried more than 10.

5

i ran out of time