Part 2

Cooper Faber

May 2021

1 Data

1.1 Default Mutex

Whomst	Run 1	Run 2	Run 3
Readers	1989078	1998152	2007972
Writers	27054	31223	27542

1.2 Fair Mutex

Whomst	Run 1	Run 2	Run 3
Readers	294156	285913	284429
Writers	112049	110068	112048

2 Discussion

2.1 Implementation

My implementation was to provide the mutex four new private variables: a reader cap, a writer cap, a writer waiting bool, and a switch (bool). If the readers ever hog the mutex too much (a value that is configurable in the program, but I left it at over 24 readers), the switch is flipped, and readers are no longer allowed in, **unless no writer is waiting**. After a certain amount of writers get through (data taken at 8, but it is also configurable), the bool gets flipped back.

2.2 Analysis

So, significantly less readers make it through (about 6x less). However, the ratio of readers to writers is much tighter, and 4x more writers manage to access the critcal section. As I outlined above, this implementation has two variables that you can change to either get closer to default behavior, or prioritze writers more. I chose the values I chose (24 and 8 for readers and writers, respectively) as the ratio of readers to writers is about 6:2. This implementation would attempt to hold the readers and writers to a similar ratio, regardless of the amount entering. However, if there are never any writers, all readers will be allowed through.