Concurrency 3

CS 444

November 22, 2018

Anton Synytsia, Eytan Brodsky, David Jansen

Contents

1 Introduction

In the following section, we provide initial guidance for the TA to evaluate our work.

2 Concurrency - Part 1

This part can basically be broken down into two other part: waiting for the list of threads to fill up, and then waiting for it to flush before allowing more threads to access the resource. To keep track of the number of threads using the resource, we use a counter. Once that counter reaches three, we set a flag called ”available” to false, which will prevent any new threads that want to access a resource from accessing it. When each thread finishes using the resource, it decrements the number of users. When the number of users becomes 0, the available flag is set to true, indicating that the list has flushed and allowing processes to start using that resource again. We implemented ”using” by having the threads sleep for a random number of seconds between 2 and 10 after unlocking the mutex. When the thread is about to decrement the number of users after it's done using the resource, it locks the mutex again to write to the num_users variable and then unlocks it. This process loops until the user decides to terminate the program.

3 Concurrency - Part 2

In the code, we have 3 threads with 3 functions. Inserters insert at the end of the list, deleters delete anywhere, and searchers just iterate through the list. Inserters must be mutually exclusive with each other and they change the size of the array, so they lock insertion_mutex and size_mutex. Deleters change the size of the list and must be mutually exclusive with each other, inserters and searchers, so they lock insertion_mutex, search_mutex, size_mutex and the flag_mutex. Flag_mutex reflects the relationship between deleters and searchers, in that searchers may operate concurrently but must cannot operate at all when a deletion is taking place. By setting the flag to 1, the program indicates that a deletion is active, making searchers wait before starting to iterate over the list. Searcher sleep from 1 to 2 seconds, deleters sleep from 1 to 5 seconds, and inserters sleep from 1 to 5 seconds as well to make sure that we can observe the program working properly. We initialize 3 inserters, 3 deleters and 3 searchers in main.