CS 415 Operating Systems

Project 3 Report Collection

Submitted to:
Prof. Allen Malony

Author:
Liam Bouffard
lbouffa6
951811278

Report

Introduction

This project consisted of 4 different parts. The first part of this project involved making a worker and a bank thread. The workers job was completing bank operation inputs and updating the value of 10 different account numbers. The different operations consisted of the following: transfer, withdrawal, deposit, and check. The transfer was formatted as such: T account_src amount account_dst; and the deposit and withdrawal was formatted as such: W/D account amount. For part1, we had only one thread (main thread) to complete the ops and to complete the banking operations. The banking thread would apply each account's reward to its balance, and then add that product to its respective balance. For part2, we only created 3 threads total, one to complete the ops, one to complete the banking operations, and the main thread. For part 3, we created many worker threads to complete 5000 transactions each. Part4 was the same as part3 except for the addition of memory mapping.

Background

In order to manage the many threads, we used barriers, locks, and cond wait. Handling the many threads was a complicated task and often led to deadlocks. I was able to work out the problems and get it working.

Implementation

My implementation was quite different from others and from what was intended. I was under the assumption we create ten worker threads, assigning each (total jobs) / (num of thread) amount of jobs. After implementing this as the base foundation for my solution, I realized we were instead supposed to assign each worker thread 5000 jobs, which would have been much easier to do. My solution also eventually gave me problems in part4 where when I tried to fork().

Performance Results and Discussion

I’m proud to announce my code does not deadlock! After realizing I approached the problem wrong (described in previous section), I came to the conclusion I'm most likely creating many less threads then I should be, meaning my program's currency will be less and take longer. My part1, part2, and part3 work perfectly as intended, but unfortunately my part4 segfaults. This is due to some questionable file pointer work I implemented, but I have yet to track down why.

Conclusion

Overall this project was a fascinating experience and a nice, but very intense, introduction to the implementation of concurrent programming. I spent most of my time on this project searching for deadlock problems to which I found print statements to be most helpful.