Report

Seth-Amittai Silvance Tisbi

1000846338

The implementation for the file locking scheme was straight forward, the Centralized Algorithm was implemented. I reused components from the last project for creating and running client server program. In this program the server creates and maintains a fixed number of threads (20 in the .jar file) that each listen to a different port (ports 2000 – 2019). The client creates and maintains a fixed number of separate processes (20 in the .jar file) that each open a socket connection to ports (2000 – 2019). The process sends its PID over the socket to the server thread that then adds this value, along with it’s own thread ID to the file and increments a count in the file. The server then sends back an acknowledgement message to the client. This is repeated a fixed number of times (3 in the .jar file) for every process. Upon completion, all threads are joined back to the main thread, all processes are waited for to die, and all connections close. A semaphore is used to prevent multiple access to the same file along with a FIFO queue that allows each requesting thread access to the shared resource (the file and counter) in the order that the request occurred, thus fulfilling a centralized locking scheme.

I learned quite a bit during this project. Prior to this assignment I had no experience with threads, processes, or semaphores in Java. I took it upon myself to try and learn and it turned out quite well, as my design is object oriented in nature and easy to follow (minus the abundance of try-catch statements). The semaphore is a basic way to implement a centralized locking scheme and the other locking methods are not a far cry from this method of implementation.

Among the issues I encountered during this assignment was learning how to manage the multiple requests coming from each process to each thread based server. Primarily, how to manage blocking and non-blocking calls that a certain function call might make. Once I read through the documentation of which calls were blocking and which were not, the implementation was fairly easy to write.