Garrett Bowser
CIS3207 Section 04
Project 3: Networked Spell Checker
Due date: 4 November 2019

Overview

This project is a server and client for a networked spell checker. The server listens for connections

over a given port and verifies that submitted words are spelled correctly. The server

can accept multiple simultaneous connections. The client executable provided with this program simulates connections from multiple clients.

Data Structures

Worker

A worker is essentially a wrapper class for a thread. A worker has associated with it an id, a thread, a boolean to indicate whether or not it is currently occupied, and a routine function. Workers are managed by a spool that passes them to their routine function so that their member variables can be accessed within it.

Spool

A spool is a collection of threads. This program uses a spool of Workers to easily initialize and manage several threads. A spool has a size, a boolean to indicate if all its threads are currently running, a vector of Workers, and a few methods to fetch information about its Workers.

Design

The server component of this program is able to operate on multiple threads simultaneously through the use of so-called locks and condition variables. When the server is run, however many threads were specified are initialized and slept until they are needed. When a new connection is made to the server, the socket (an integer) is pushed into a fixed-size queue and a thread is awoken. The thread will service the socket until the socket disconnects. Locks are used to ensure that the queue is only accessed at appropriate times. For example, the thread must be locked when it “pops” a socket off the queue, because it is possible that the thread can be preempted between pulling a value off the queue and removing the same value from the queue, resulting in (potentially) the same value being assigned to multiple threads. A similar problem is addressed by the logging thread. The logging thread receives jobs from locked Workers and writes them to a file. This design pattern must be used, because allowing multiple threads to write to a file without a method of synchronization can have *catastrophic* consequences.

The client for this program operates by essentially the same principles. Multiple threads are created that make connections to a running server. The words that are sent are logged and synchronized via the same method.