Yasmine Watkins
CIS 3207 -001
Networked Spell Checker Program Writeup

Program Description

The Networked Spell Checker is a server that can accept multiple command-line arguments designed to connect to clients in order to spell check words entered in by a user. The program uses a default dictionary file(word.txt) and provides feedback of “OK” if a word is spelled correctly or “MISSPELLED” if a word is spelled incorrectly. All words entered in after connection to the server is made will be printed into a logfile (“log.txt). The logfile will also include data such as the time the spell check of a word was requested and completed in seconds and microseconds along with the feedback described above.

Program Design

There are two main files for this program: server.c and server.h. Server.h is the header file that includes two structs, socketBuff and logBuff. The first, socketBuff, is a circular buffer which is used to store client connection descriptors as well as their priority as structs. The logBuff has similar functionality but it will store an array of strings that will be printed to the logfile. If the buffers become full, the put() function will use the pthread cond variable to wait until the get () function removes a client_socket from the buffer, with the opposite being true as well. When put() and get() try to access the socketBuff, they will be pthread locked until finished. This file also includes function declarations and definitions.

Server.c includes the worker thread, log thread, dictionary and the put and get functions. The worker thread(worker_thread) holds threads that won’t stop working until the program terminates. It uses the client_sockect to read the word that has been entered to be spellchecked by the client. Once that is complete the data described above will be printed into the logfile. The log thread(log_thread) will use the data added by the logBuff to write the logfile. The dictionary function is used to handle the dictionary file.

Testing, Results and Debugging

I first tested to make sure that the right number of threads were created and that they were able to print a line. My results were as expected with 5 different lines printing as defined with THREAD_NUM 5. My second test was to check the functionality of my circular buffers. My third test was to check that the default dictionary was functionally properly by adding in my word.txt file. I checked that the program would be able to recognize correctly spelled and misspelled words. I did find that there was an error in my results which was that although it correctly spellchecked a word, the last character of each word was missing. For example, neighborhood was being displayed as neighborhoo. I also found that the message that was being displayed once connected to the server was too short, so the full message was not being displayed.

Discussion and Analysis

It was clear that the use of multiple threads when working with a number of clients was of high importance. This is because if not done so, jobs will be forced to wait which makes the program inefficient. I also learned that concurrency allows for data to be processed from multiple sources with the help of sockets as well. While testing/debugging to I was able to see firsthand the differences between the main thread and worker thread, as well as how the use of different buffer and max thread counts impacts performance differently.