

Network Spelling Checker Unit Testing.

Abraham Schultz
3207 lab3

Introduction

For testing here, I took a modular approach and tested each part of the project as an individual unit.

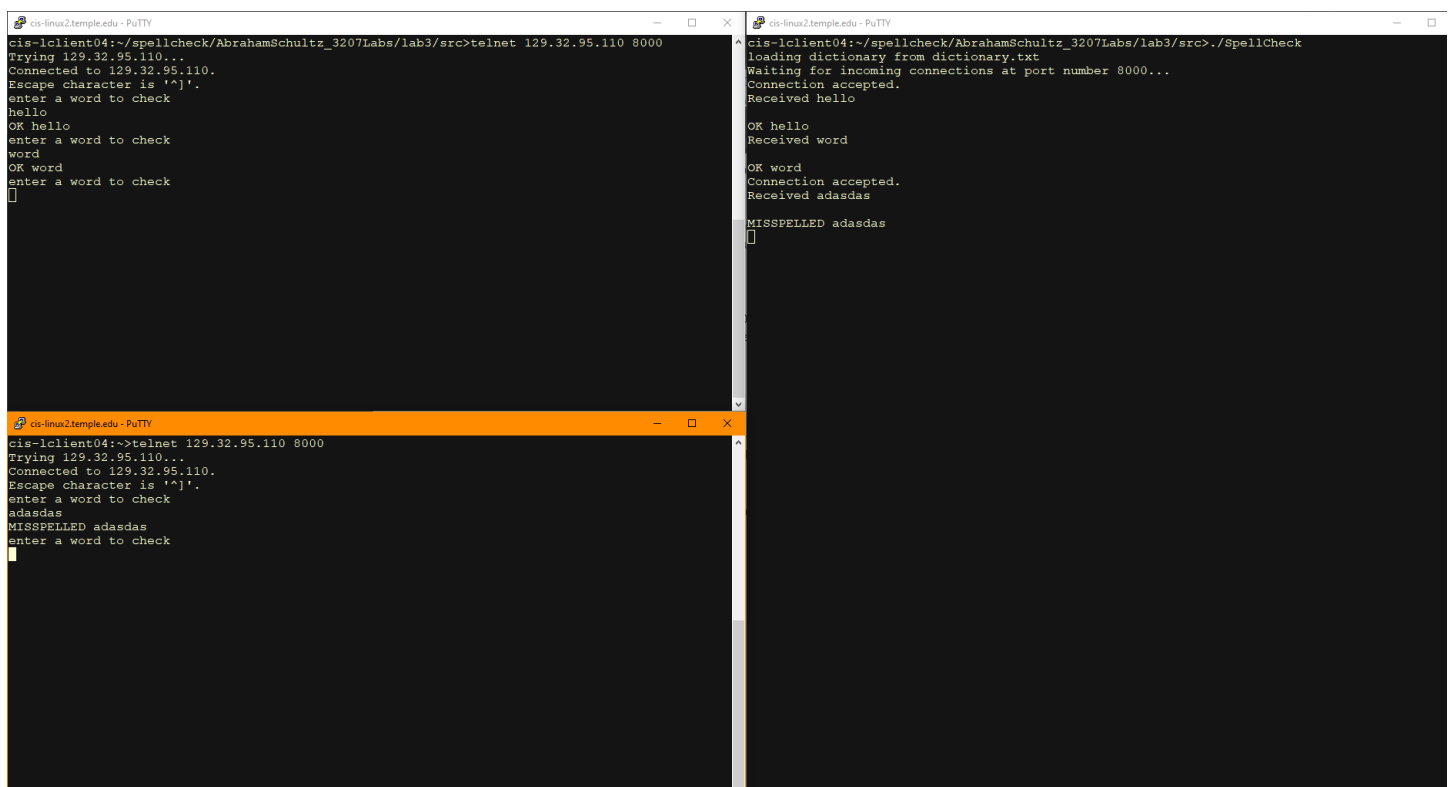
Unit Test: Dictionary

The dictionary unit test was simple enough. I just picked words at random from the dictionary file and checked that my program could correctly identify them or not. I also printed out to the screen words as they were loaded in the array data structure to make sure we were loading the words correctly. A more thorough approach would have been to check every word using the dictionary file itself as the input to be checked.

If it were to be improved, I would have implemented case sensitivity to the spell checker. But the design specifications did not call for this.

Unit Test: network Connection

Below is a screen shot from the server being used on clinet04 of temples servers. The upper right-hand corner is the server accepting words and checking them. Each of the other servers are sending words to check.



```
cis-lclient04:~/spellcheck/AbrahamSchultz_3207Labs/lab3/src>telnet 129.32.95.110 8000
Trying 129.32.95.110...
Connected to 129.32.95.110.
Escape character is '^]'.
enter a word to check
hello
OK hello
enter a word to check
word
OK word
enter a word to check
^

cis-lclient04:~/spellcheck/AbrahamSchultz_3207Labs/lab3/src>./SpellCheck
loading dictionary from dictionary.txt
Waiting for incoming connections at port number 8000...
Connection accepted.
Received hello
OK hello
Received word
OK word
Connection accepted.
Received adadas
MISSPELLED adadas
^

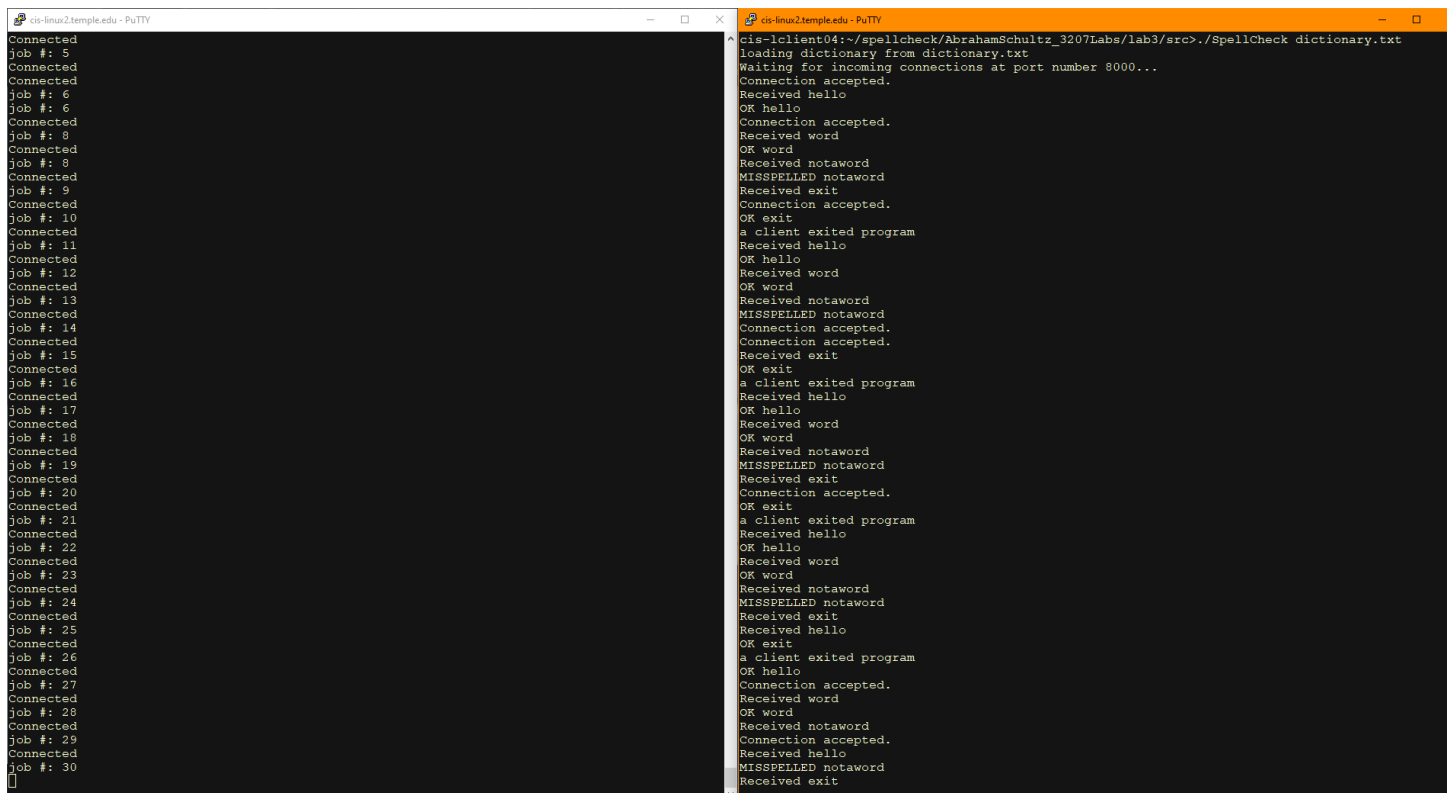
cis-lclient04:~/spellcheck/AbrahamSchultz_3207Labs/lab3/src>telnet 129.32.95.110 8000
Trying 129.32.95.110...
Connected to 129.32.95.110.
Escape character is '^]'.
enter a word to check
adadas
MISSPELLED adadas
enter a word to check
^
```

(image: server with 2 clients)

Unit Test work Queue

This is harder to test, because its hard to prove your mutual exclusion is working. And it is much easier to disprove that is. To that end I attempted to break the mutual exclusion by having multiple clients connecting as close as possible in time as I could. I also used strategically placed print outs to keep track of what was in the queue. In the screen shot

below I used a multithreaded client program to connect an arbitrary amount of clients who would connect send a few words then exit, making room for the next client.

The image shows two side-by-side terminal windows. The left window, titled 'cis-linux2temple.edu - PuTTY', displays a list of 30 'Connected' messages, each followed by a 'job #' ranging from 5 to 30. The right window, also titled 'cis-linux2temple.edu - PuTTY', shows the server's internal processing for each client. It starts with 'loading dictionary from dictionary.txt' and 'Waiting for incoming connections at port number 8000...'. For each client, it logs 'Connection accepted.', the received message (e.g., 'hello', 'word', 'notaword', 'misspelled notaword', 'exit'), and the response ('OK'). It also notes when a client exits with 'a client exited program'.

(image : server with 30 clients)

Unit Test : log Queue

This again was difficult to test without having a truly multithreaded test client. Or having multiple people who can help test. With one person and no multithreaded test client queries for spelling do not arrive at the same time. So of course, the log queue appears to work correctly. Below is a printout from the log.txt file after running a multithreaded test client.

cis-lclient04:~/spellcheck/AbrahamSchultz_3207Labs/lab3/src>cat log.txt

OK hello

OK hello

OK word

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK hello

OK word

MISPELLED notaword

OK word

OK hello

MISPELLED notaword

OK word

OK hello

MISPELLED notaword