ReadMe

Tuan Nguyen

Network Spell checker

The project is created a spell checker server to connect with multiple of clients. Indeed, gain experience with multi-threaded programming and the synchronization problem. First, I have a server main thread to take command line argument (name of dictionary file). If dictionary file not provided use the default dictionary. The program should take an argument as a port number which to listen for incoming connection. The server has two functions: one to accept and distribute connection request from client, anther to contract a log file of all spell check. When the server start, take in command line argument in the form ./serv port dict\_name - If port or dict\_name unspecified, use the default one - When server start, it load in the dictionary to an array - create socket descriptor to connect it with clients and push the socket to a queue - create a threadPool consists of multiple thread workers which will handle the jobs - create another separate thread to handle writing to logfile. I also have a worker thread to Deque the socket descriptor from the queue. Beginning read in words from user clients - check if match, echo the result back to both server and clients - push the results to the logqueue

Main.c

-do all the jobs of server and workers

logqueue.c

-contains the message results

utils.c

-char \*dict\_load(char \*name): load the dictionary into an array -int open\_listenfd(char \*port): create listening socket and return it -ssize\_t readLine(int fd, void \*buffer, size\_t n): read in user input to a buffer

queue.c

-The work queue which will contain the socket descriptor, because multiple threads can update it at the same time. We need to have mutex and condition variable to synchronize the read and write

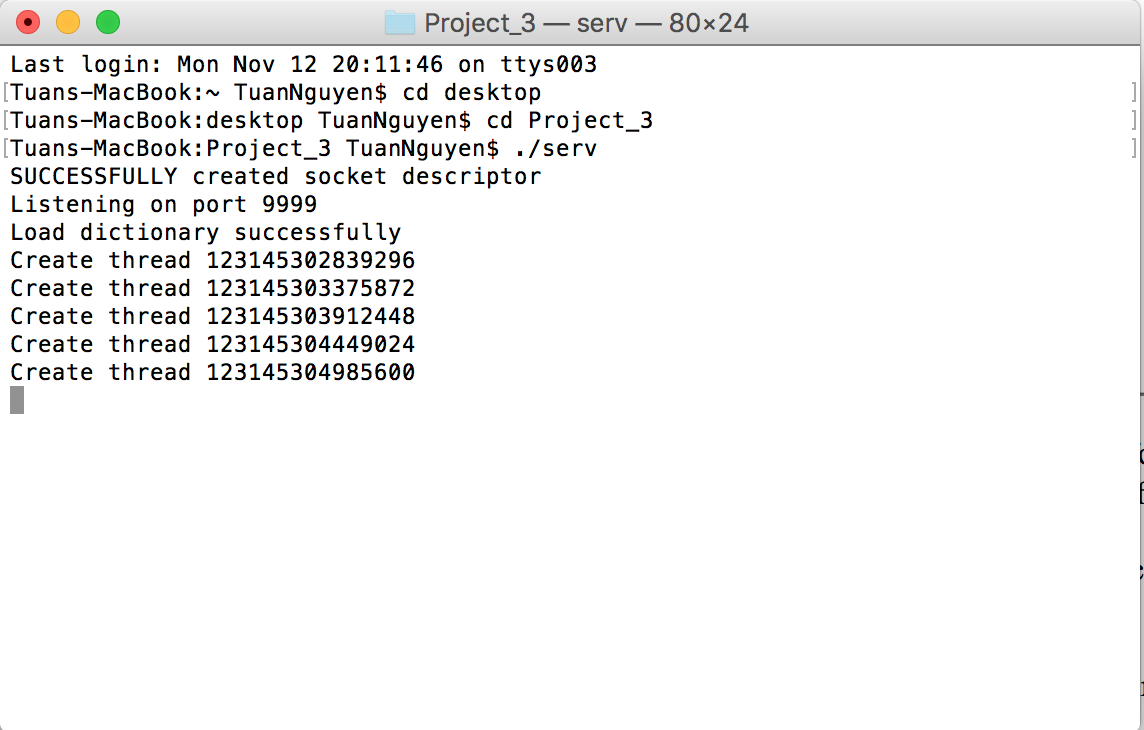
log.txt

-have line of word that misspelled and correct spell for worker to spell check

testing

-make

-./serv

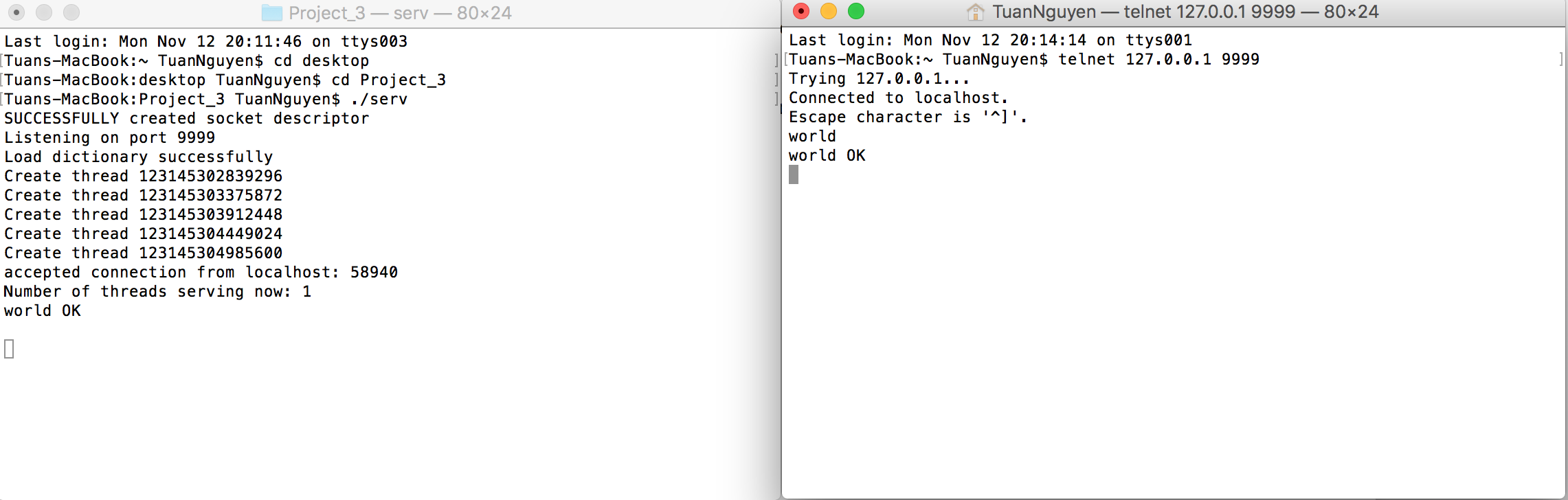


Notes: Default port is 9999 and maximum number of thread workers is 5

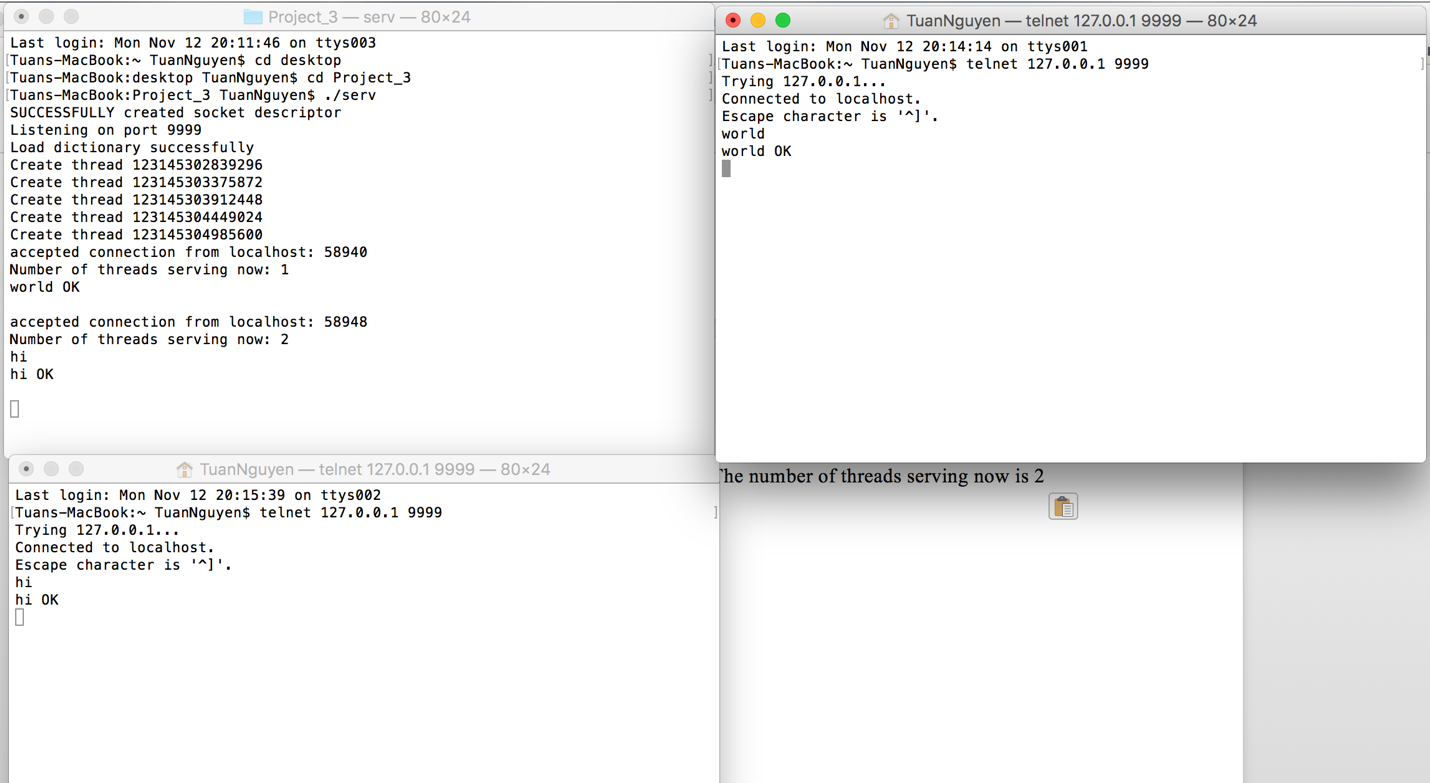
We start by creating the server, it also creates the socket descriptor, load in the dictionary and create 5 threads.

Then we create 1 client connect to that server, one thread will start serving that client

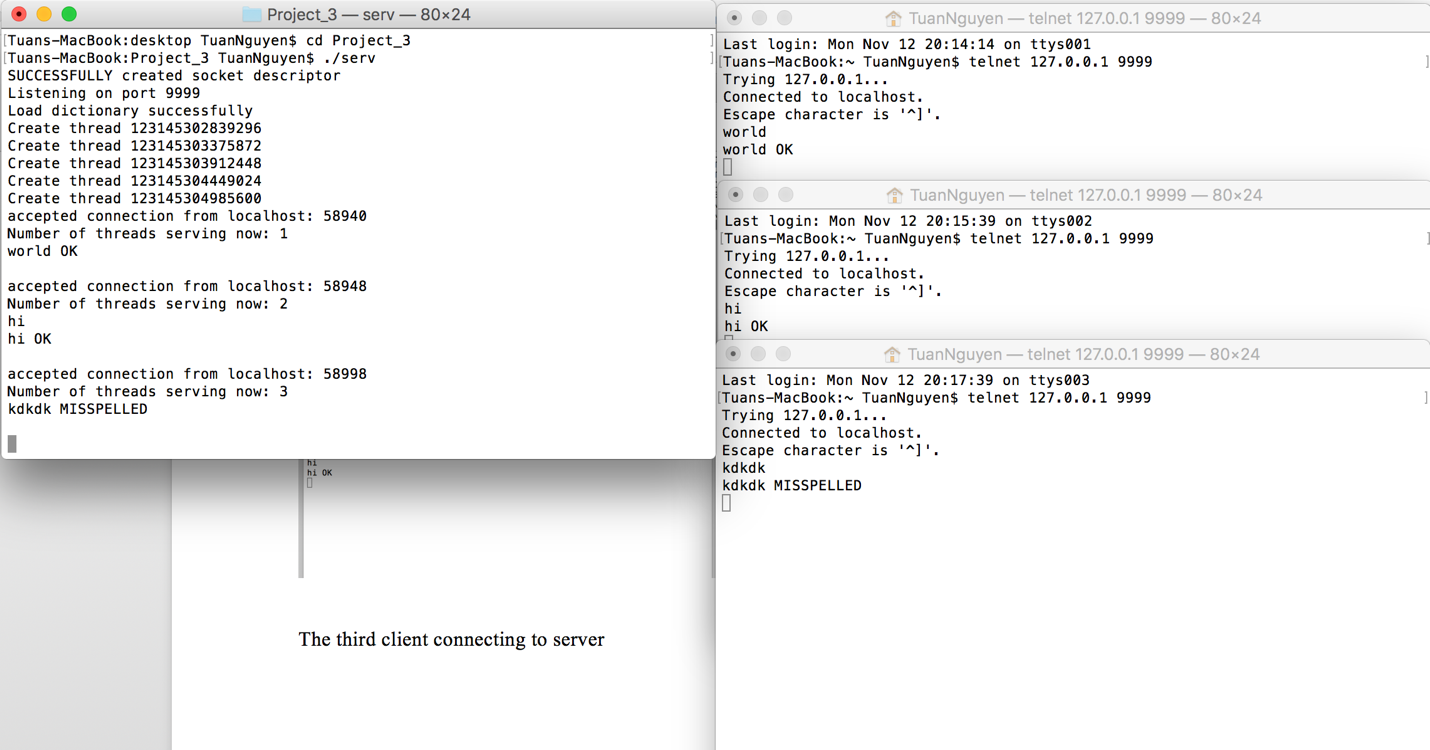
Try some simple test case to test the spellchecker:



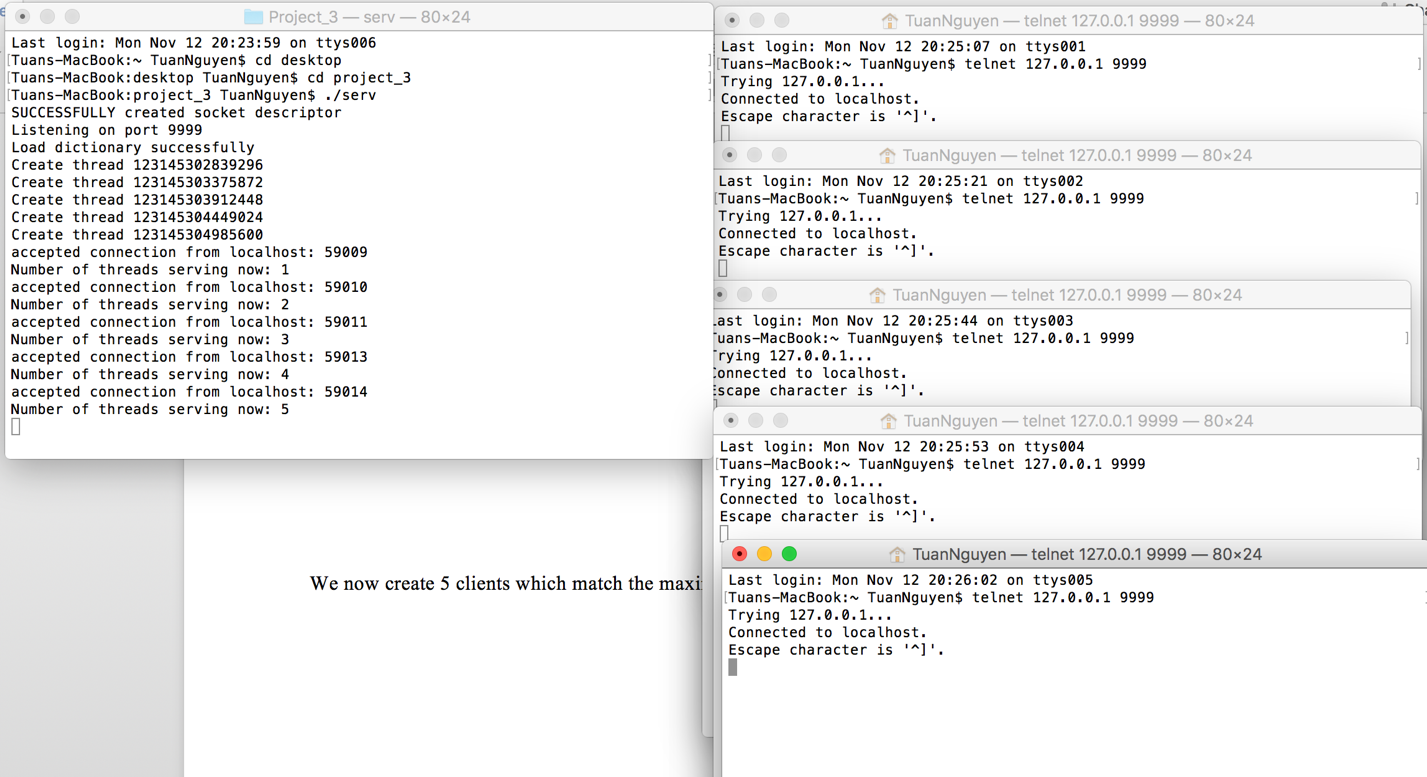
We open another client connecting to the server. The number of threads serving now is 2



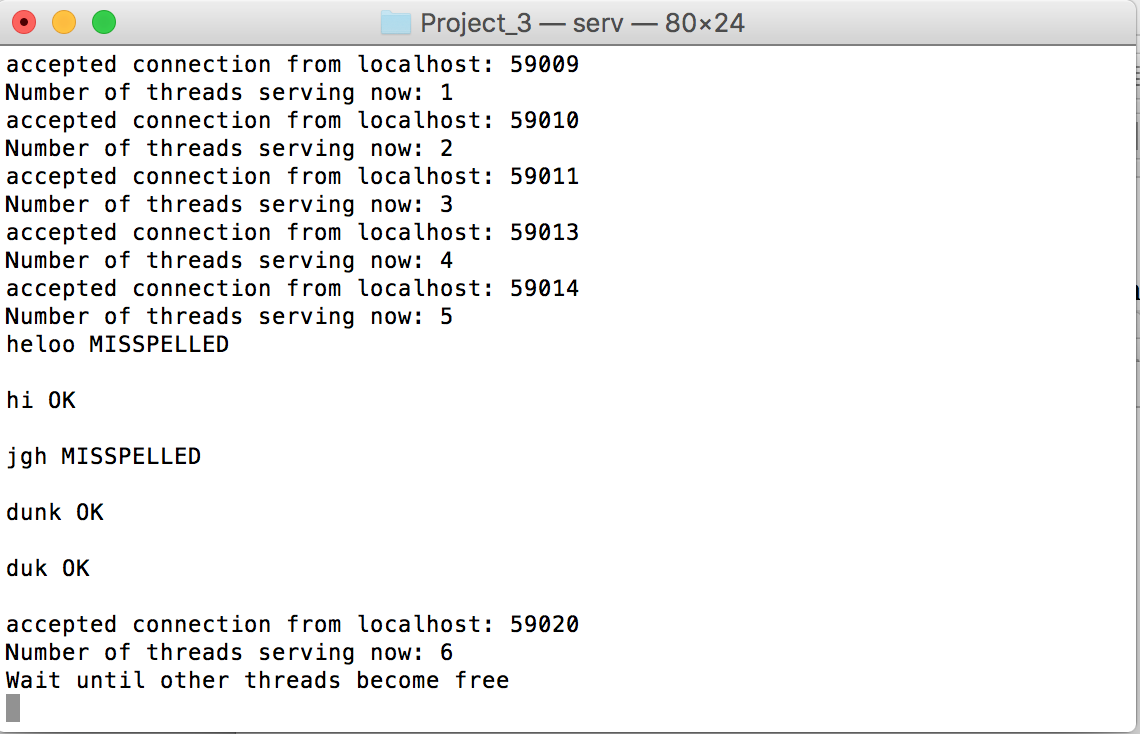
The third client connecting to server



We now create 5 clients which match the maximum number of threads the server created



When we create 6 clients, the server forces it to wait until a serving thread is available



When a client was closed, the server displays a connection closed message and start serving the waiting client

Write the result into log.txt

