**Ishan Patel**

**CIS 3207**

**PROJECT 3**

**Networked Spell Checker**

**Introduction:**

Spell checkers are useful utility programs often bundled with text editors. In this assignment, you'll create a networked spell check server. The purpose of the assignment is to gain some exposure and practical experience with multi-threaded programming and the synchronization problems that go along with it, as well as with writing programs that communicate across networks.

Spell check server is to be a process that will read sequences of words. If a word is in its dictionary, it's considered to be spelled properly. If not, it's considered to be misspelled. The dictionary itself is nothing but a very long word list stored in plain text form.

When the server starts, the main thread opens the dictionary file and reads it into some data structure accessible by all of the threads in the program. It also creates a fixed-sized data structure which will be used to store the socket descriptors of the clients that will connect to it. This data structure will also be accessible to all threads. The main thread creates a pool of NUM\_WORKERS worker threads, and then immediately begins to accept and distribute connection requests.

The work queue is a shared data structure, with the main thread acting as a producer, adding socket descriptors to the queue, and the worker threads acting as consumers, removing socket descriptors from the queue. Similarly, the log queue is a shared data structure, with the worker threads acting as producers of results into the buffer and a server log thread acting as a consumer, removing results from the buffer. Because we have concurrent access to these shared data structures, we must synchronize access to them using the techniques that we've discussed in class so that: 1) each client is serviced, and 2) the queues do not become corrupted.

No more than a single thread at a time may manipulate the work queue. We've seen that this can be guaranteed through the proper use of mutual exclusion. Your solution should include attempts at synchronization using locks and condition variables. No more than one worker thread at a time should manipulate the log queue at any one time. This can be ensured through the proper use of mutual exclusion. Again, attempts at synchronization should be using locks and condition variables.

**Server's network address will be the loopback address of 127.0.0.1**

Use the UNIX telnet client, which, in addition to running the telnet protocol, can be used to connect to any TCP port.

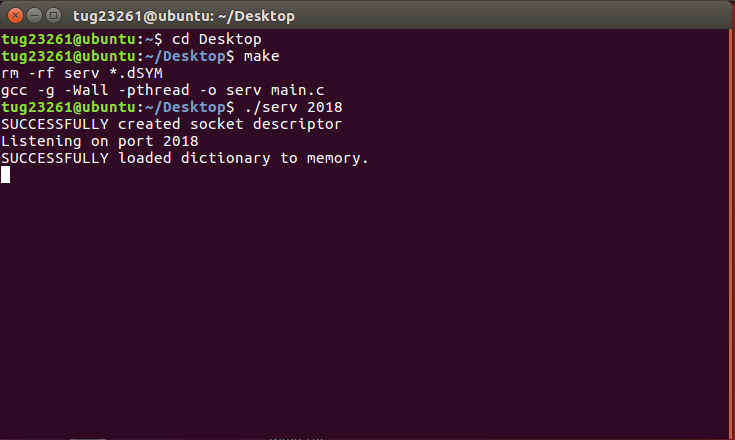
**Running the program:**

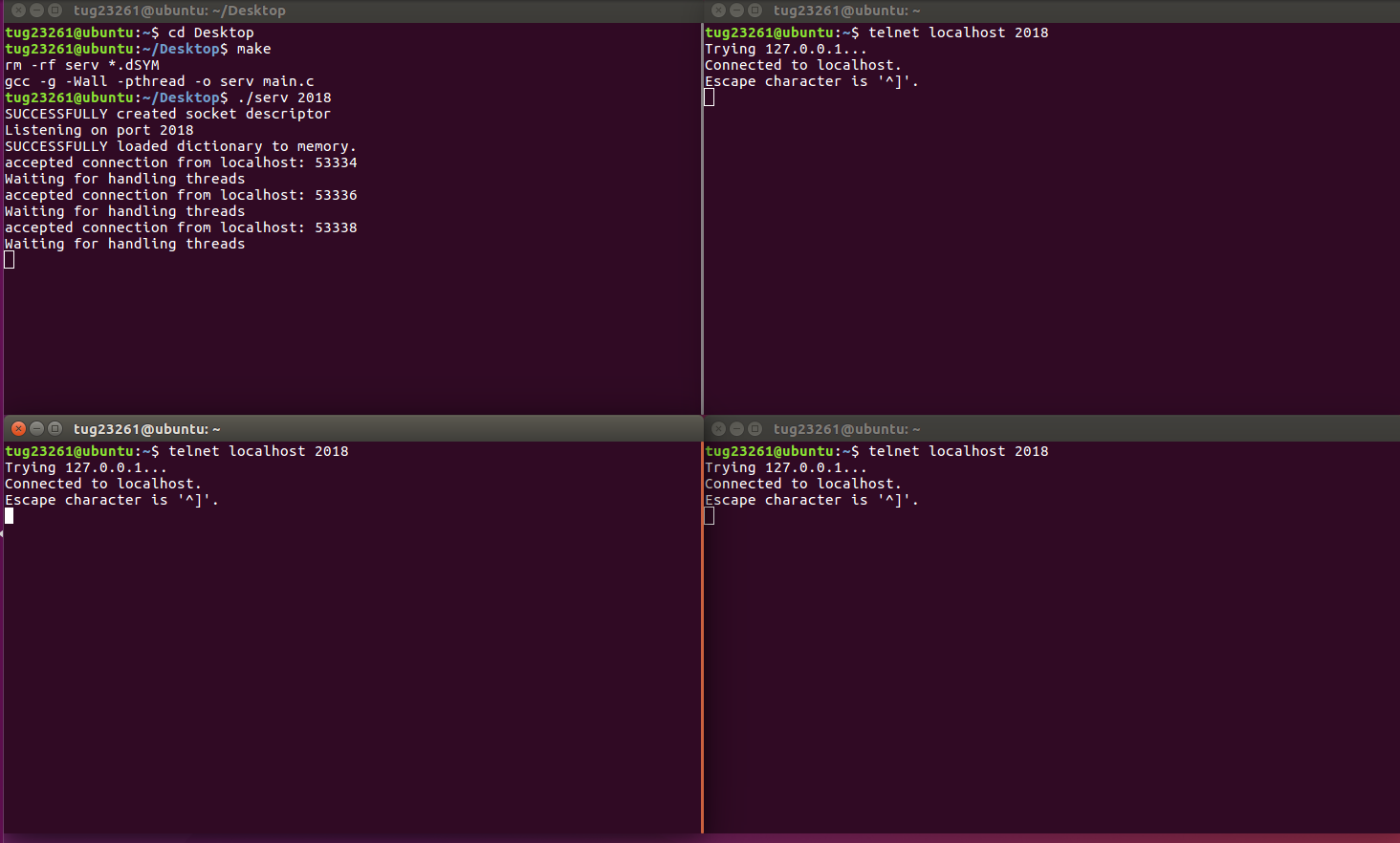
This program is set to run in Linux environment with c99 standard. It has a make file which simplifies and saves time in inputting long commands in the command prompt. Just open the command prompt in linux and go to the folder where you placed the files using the ‘cd’ command. Then enter in ‘make’ and then **‘./serv [Port Number] [Dictionary file]’**. If user does not enter the Port Number and Dictionary file then the program uses the **default port 9999** and **default dictionary “words.txt”**. if user just enters a port number than the program uses the default dictionary file.

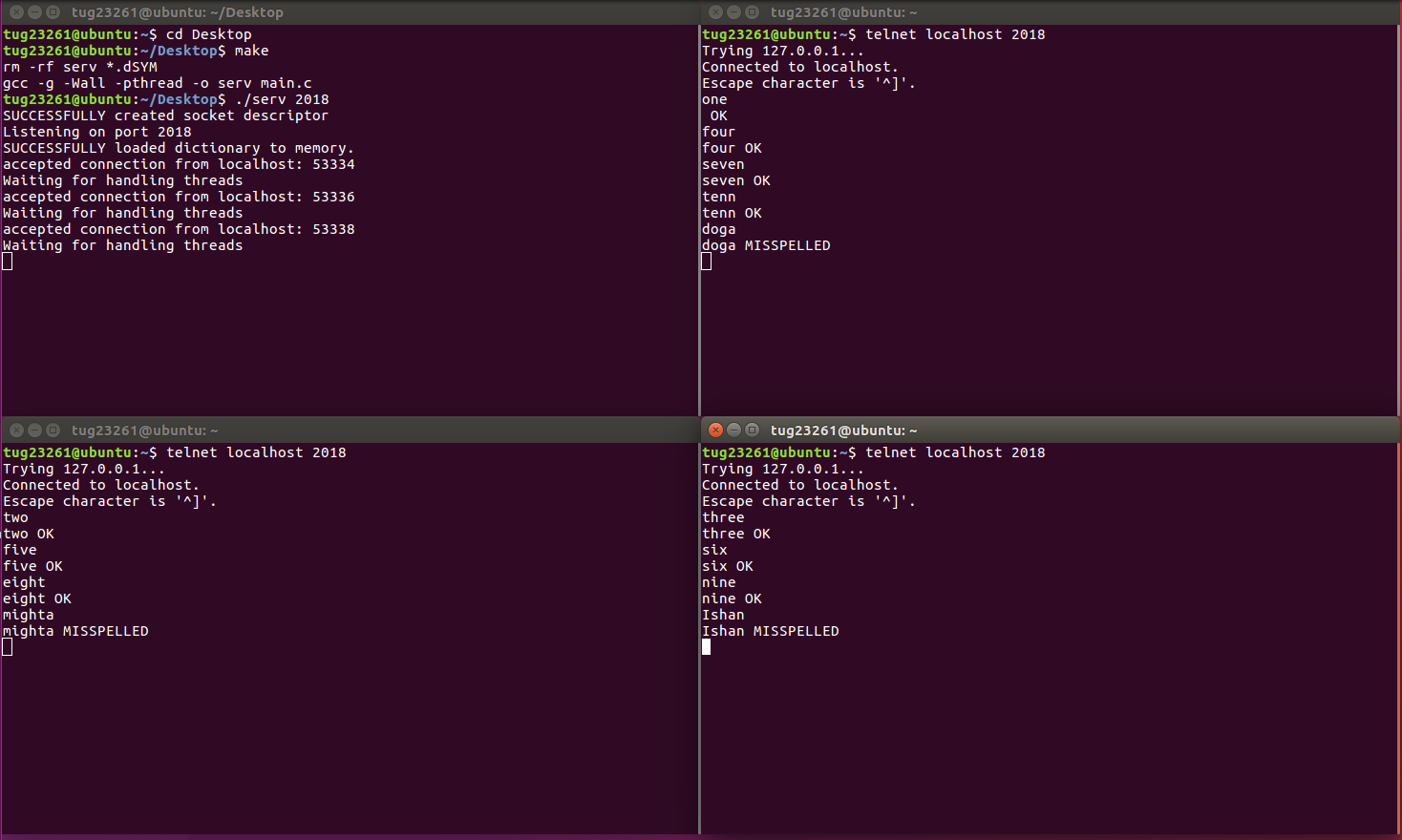
Doing this would run the server with the dictionary loaded in it.

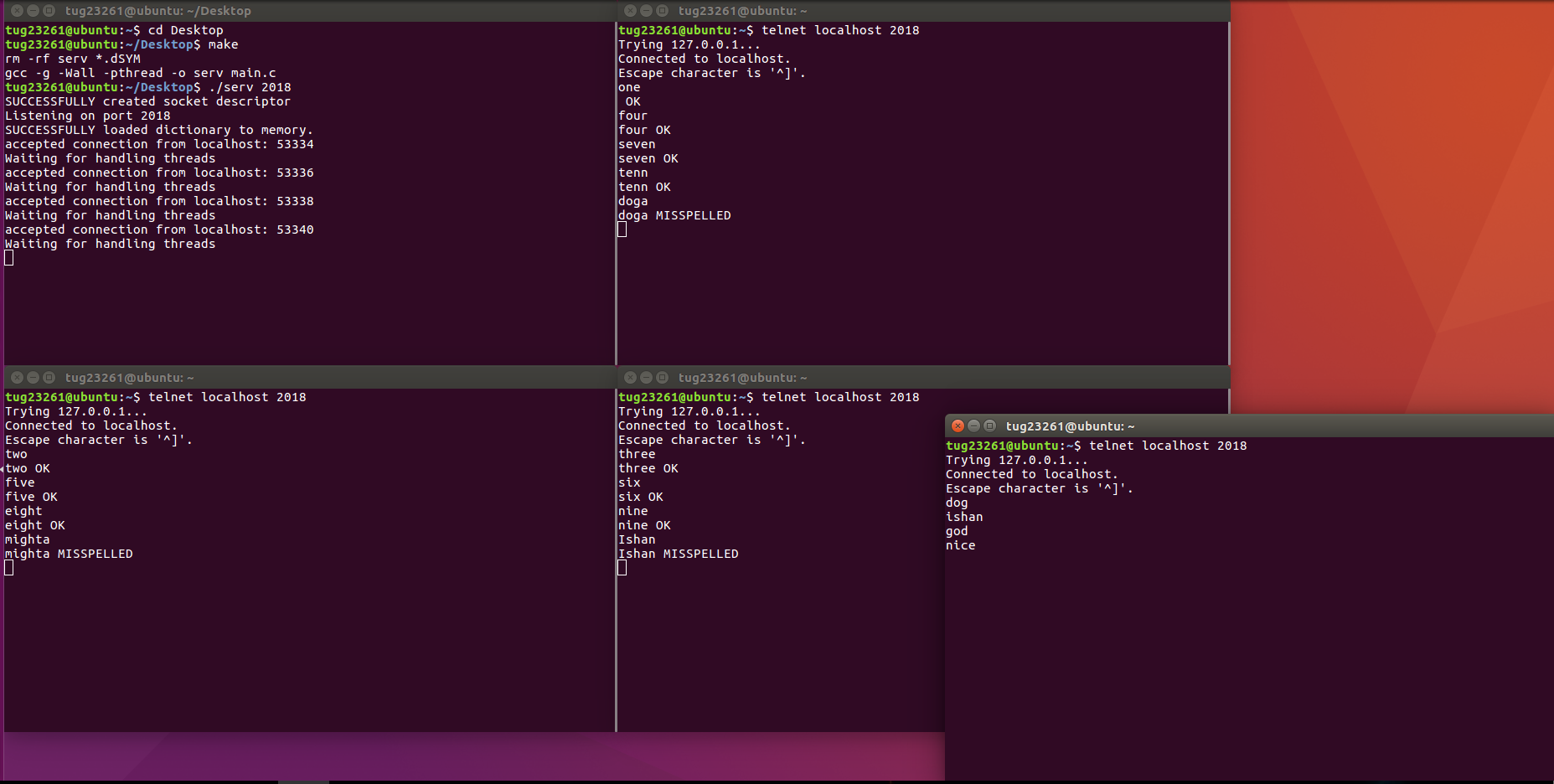
Now, for clients in different terminals you can host the clients by typing in “telnet localhost [port Number]”, where the port number is the one where the server is hosting.

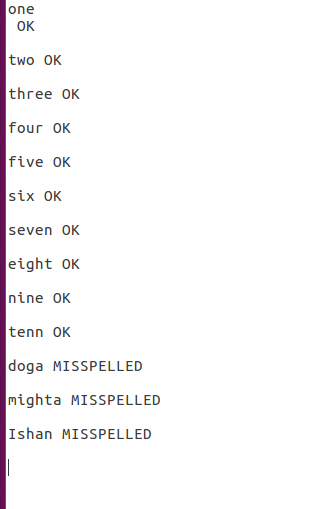
**Testing the Server:**

****

****

****

****

****