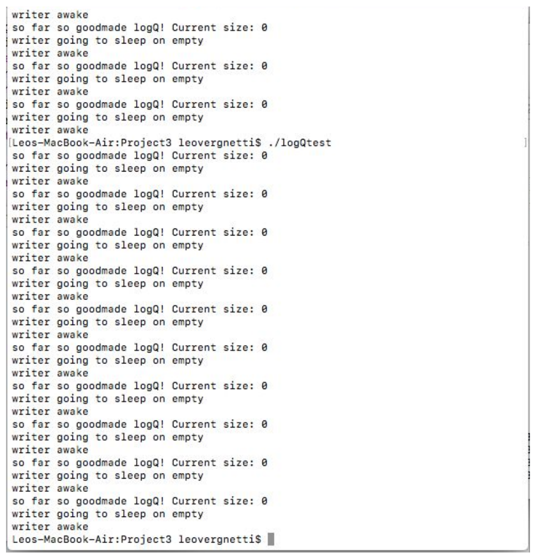
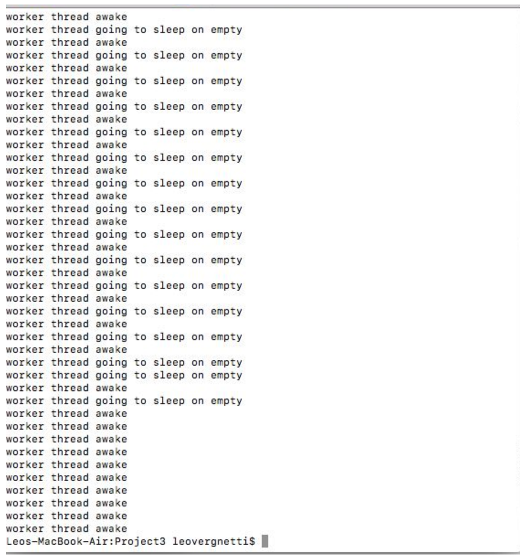
Leo Vergnetti
Project 3

logQ.c Tests and Descriptions
logQ.c holds the queue in which strings are stored before written to the logfile by the writer thread. The logQ was adapted from project 1's cpuQ, made threadsafe by use of condition variables and mutex lock. It is a linked list, and has no maximum value. If the logQ is empty, the writer thread sleeps until being signalled by a worker thread upon insertion. The server thread can also signal the logQ using serverFinished. This only occurs when the server is finished accepting connections.
The test for the logQ is in the logQtest.c file. The test creates 10 worker threads, and each thread proceeds to add a string (From a predefined array to the logQ). The writer thread pops the values from the screen. The file passes, if for each instance the code exits normally. If errors are present (UNWRITTEN VALUES in particular) logQ fails. The result is below:



```
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
Leos-MacBook-Air:Project3 leovergnetti$ ./logQtest
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
so far so goodmade logQ! Current size: 0
writer going to sleep on empty
writer awake
Leos-MacBook-Air:Project3 leovergnetti$
```

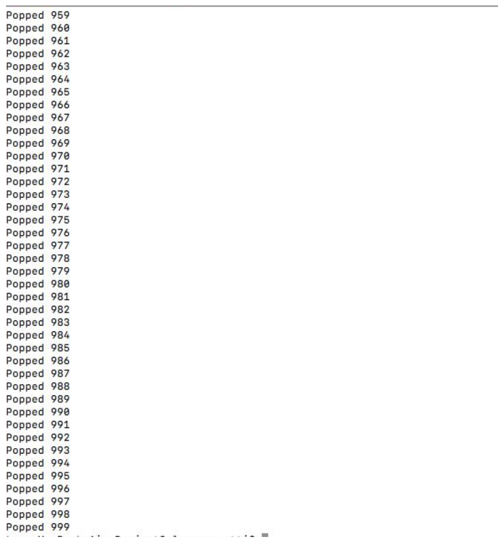
jobQ.c Tests and Descriptions
jobQ.c holds the queue in which jobs are placed by the server. It is almost identical to logQ, except for its finish methods. The server pushes integers into the jobQ. These integers are then popped by the workerthreads, who sleep if the jobQ is empty. Upon completion, the server calls serverFinished, broadcasting that it will no longer be accepting any connections. The workers continue to pop values from the jobQ until it is empty at which point the threads complete. Upon joining, the server thread destroys the Q.
The test demonstrates this cycle located in jobQtest.c. as with logQ, we the test succeeds when program exits normally. We see the output below.



```
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread going to sleep on empty
worker thread awake
worker thread going to sleep on empty
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
worker thread awake
Leos-MacBook-Air:Project3 leovergnetti$ 
```

logQ.c Tests and Descriptions
logQ.c holds the queue in which

logQ.c and jobQ.c Interaction Tests and Descriptions
workertest.c tests the worker threads ability to pull from the jobQ and push to the logQ. The test simply shows each thread, which prints as it sleeps or wakes. A pass on workerthread.c should show each value from 1 to 999 output to the screen, and the process should exit normally. The (end) output is below

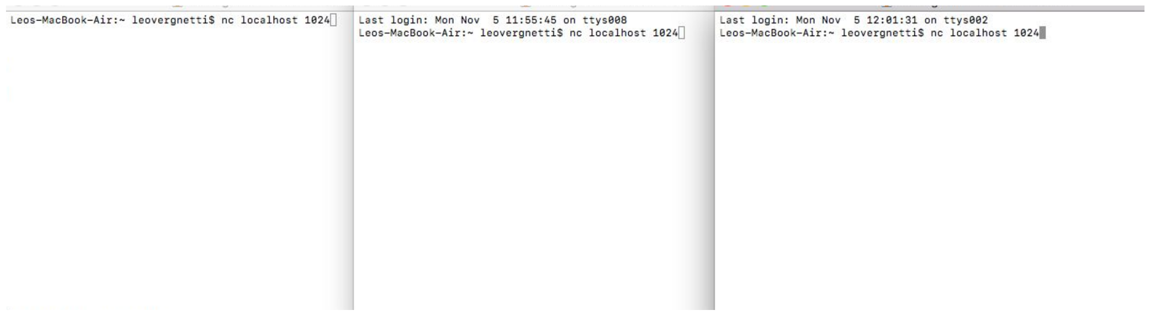


```
Popped 959
Popped 960
Popped 961
Popped 962
Popped 963
Popped 964
Popped 965
Popped 966
Popped 967
Popped 968
Popped 969
Popped 970
Popped 971
Popped 972
Popped 973
Popped 974
Popped 975
Popped 976
Popped 977
Popped 978
Popped 979
Popped 980
Popped 981
Popped 982
Popped 983
Popped 984
Popped 985
Popped 986
Popped 987
Popped 988
Popped 989
Popped 990
Popped 991
Popped 992
Popped 993
Popped 994
Popped 995
Popped 996
Popped 997
Popped 998
Popped 999
```

Server.c Description

server.c contains all logic for the server. Through 5 main methods. It uses initServer(), runServer(), destroyServer(), workerThr(), and writerThr(). The initServer method first checks the args to see whether the user supplied a port number or to use defaults. It then checks if the user supplied a dictionary or if it should use default dictionary ("words.txt" in this case). Note, dictionary methods are in a separate dictionary.c file. (initDictionary, destroyDictionary, and lookupWord). Upon proper initialization, the server opens a listening socket (using open_listenfd.c supplied by lab instructor), and upon successful accept, pushes the connected socket to the jobQ. Worker threads are responsible for removing this socket, writing and reading the word from it, and replying to the user. The writer thread pops values from this logQ, and writes them to the logFile. (NOTE, in order to keep track of which thread serviced which, an idFactory is used to initialize each worker thread with an id number. Since multiple threads access it is protected by a mutex idLock.) There were several tests associated with this, that were already verified (by jobQ , logQ and workerThread tests).

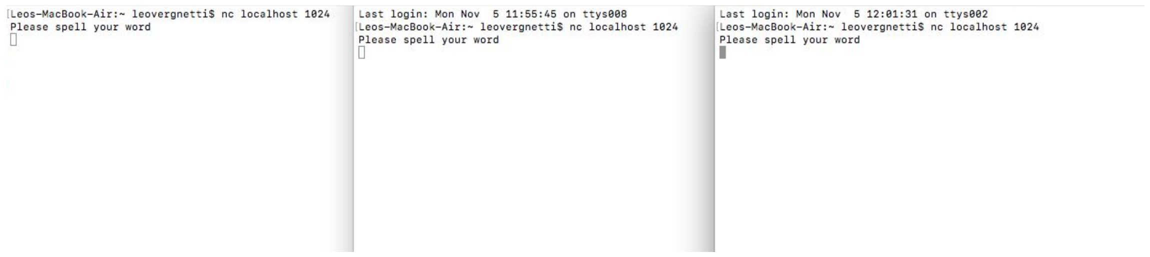
The server is tested using netcat, where we connect several times from separate terminal windows. In each case the logfile verifies correctness of the multithreaded log writing, and the ability to handle simultaneous connections. The worker id is printed in the logfile, the following images demonstrate.



```
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
```

```
Last login: Mon Nov  5 11:55:45 on ttys008
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
```

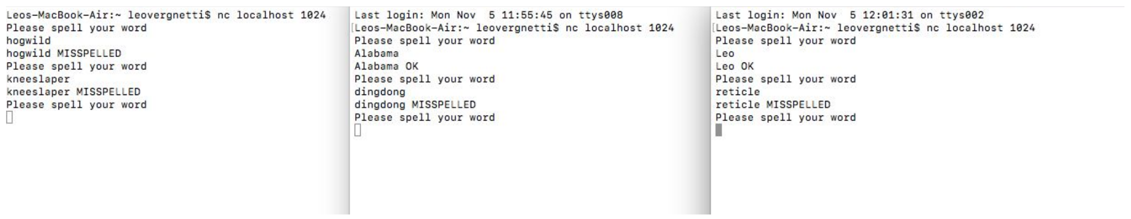
```
Last login: Mon Nov  5 12:01:31 on ttys002
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
```



```
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
```

```
Last login: Mon Nov  5 11:55:45 on ttys008
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
```

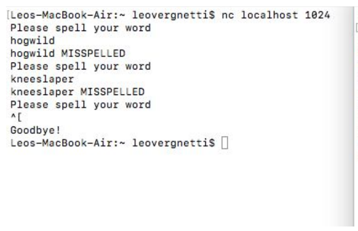
```
Last login: Mon Nov  5 12:01:31 on ttys002
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
```



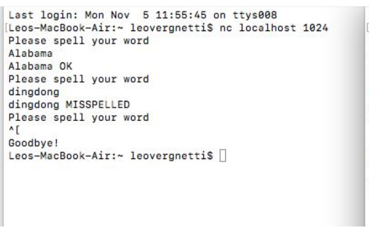
```
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
hogwild
hogwild MISSPELLED
Please spell your word
kneeslaper
kneeslaper MISSPELLED
Please spell your word
```

```
Last login: Mon Nov  5 11:55:45 on ttys008
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
Alabama
Alabama OK
Please spell your word
dingdong
dingdong MISSPELLED
Please spell your word
```

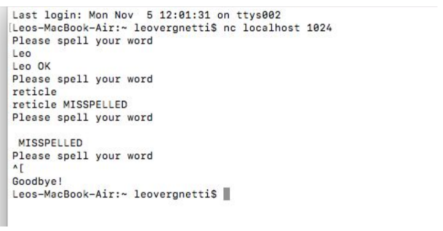
```
Last login: Mon Nov  5 12:01:31 on ttys002
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
Leo
Leo OK
Please spell your word
reticle
reticle MISSPELLED
Please spell your word
```



```
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
hogwild
hogwild MISSPELLED
Please spell your word
kneeslaper
kneeslaper MISSPELLED
Please spell your word
^[
Goodbye!
Leos-MacBook-Air:~ leovergnetti$
```

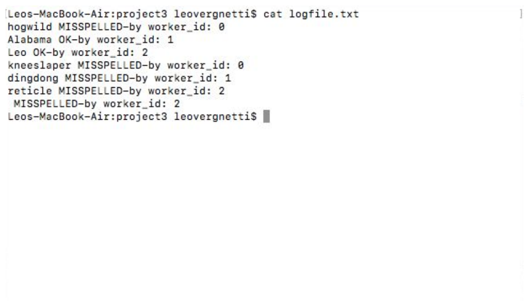


```
Last login: Mon Nov  5 11:55:45 on ttys008
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
Alabama
Alabama OK
Please spell your word
dingdong
dingdong MISSPELLED
Please spell your word
^[
Goodbye!
Leos-MacBook-Air:~ leovergnetti$
```



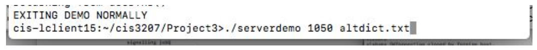
```
Last login: Mon Nov  5 12:01:31 on ttys002
Leos-MacBook-Air:~ leovergnetti$ nc localhost 1024
Please spell your word
Leo
Leo OK
Please spell your word
reticle
reticle MISSPELLED
Please spell your word

 MISSPELLED
Please spell your word
^[
Goodbye!
Leos-MacBook-Air:~ leovergnetti$
```

And the logfile from the server shows:

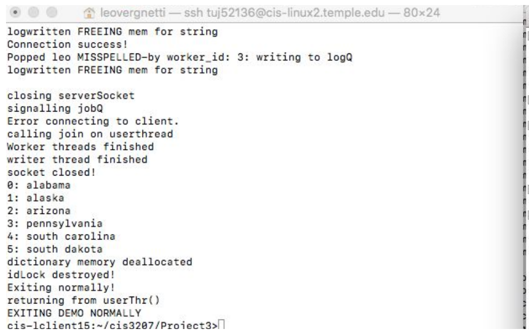


```
Leos-MacBook-Air:project3 leovergnetti$ cat logfile.txt
hogwild MISSPELLED-by worker_id: 0
Alabama OK-by worker_id: 1
Leo OK-by worker_id: 2
kneeslaper MISSPELLED-by worker_id: 0
dingdong MISSPELLED-by worker_id: 1
reticle MISSPELLED-by worker_id: 2
 MISSPELLED-by worker_id: 2
Leos-MacBook-Air:project3 leovergnetti$
```

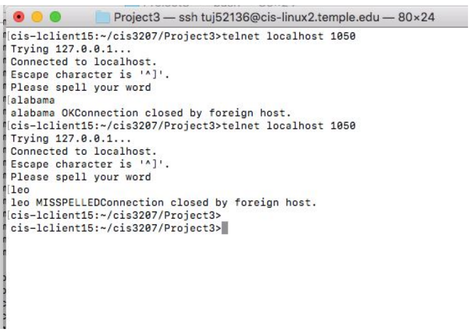
We also test the loading of an alternate dictionary and misspelled words in the next group. First we specify altdict.txt to the server, with non-default portnumber. We then connect via telnet, the result is shown below.



```
EXITING DEMO NORMALLY
cis-lclient15:~/cis3207/Project3>./serverdemo 1050 altdict.txt
```



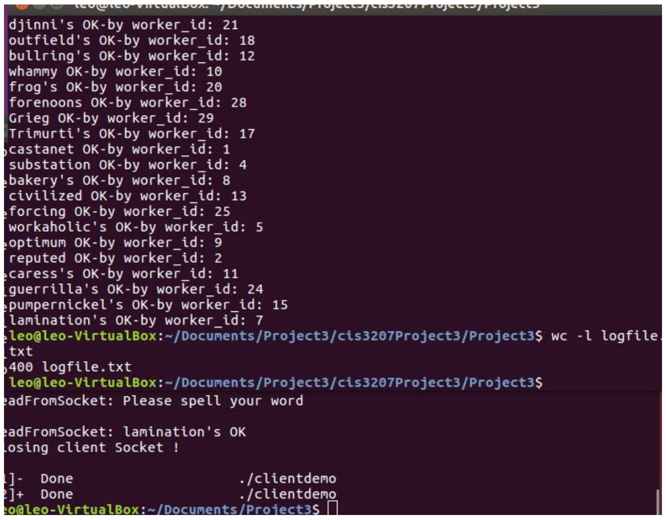
```
logwritten FREEING mem for string
Connection success!
Popped leo MISSPELLED-by worker_id: 3: writing to logQ
logwritten FREEING mem for string

closing serverSocket
signalling jobQ
Error connecting to client.
calling join on userthread
Worker threads finished
writer thread finished
socket closed!
0: alabama
1: alaska
2: arizona
3: pennsylvania
4: south carolina
5: south dakota
dictionary memory deallocated
idLock destroyed!
Exiting normally!
returning from userThr()
EXITING DEMO NORMALLY
cis-lclient15:~/cis3207/Project3>
```



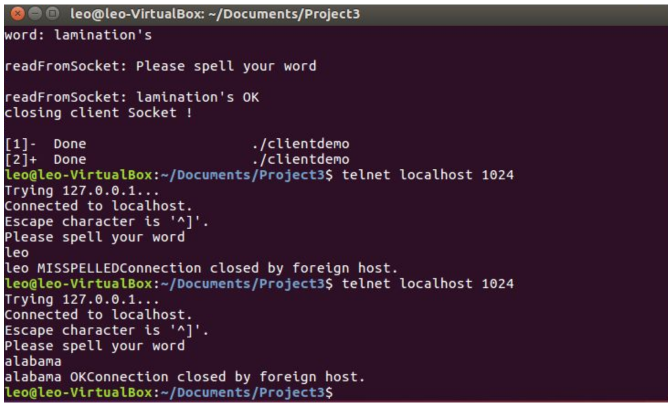
```
cis-lclient15:~/cis3207/Project3>telnet localhost 1050
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Please spell your word
alabama
alabama OKConnection closed by foreign host.
cis-lclient15:~/cis3207/Project3>telnet localhost 1050
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Please spell your word
leo
leo MISSPELLEDConnection closed by foreign host.
cis-lclient15:~/cis3207/Project3>
cis-lclient15:~/cis3207/Project3>
```

NOTE*

Clientdemo uses arc4random_uniform to choose the random word to send to the server. The unix test for clientdemo was thus omitted from the above tests, however, it can be done using -lbsd from the terminal command, with similar output. The remainder is shown below.



```
djinni's OK-by worker_id: 21
outfield's OK-by worker_id: 18
bullring's OK-by worker_id: 12
whammy OK-by worker_id: 10
frog's OK-by worker_id: 20
forenoons OK-by worker_id: 28
Grieg OK-by worker_id: 29
Trimurti's OK-by worker_id: 17
castanet OK-by worker_id: 1
substation OK-by worker_id: 4
bakery's OK-by worker_id: 8
civilized OK-by worker_id: 13
forcing OK-by worker_id: 25
workaholic's OK-by worker_id: 5
optimum OK-by worker_id: 9
reputed OK-by worker_id: 2
caress's OK-by worker_id: 11
guerrilla's OK-by worker_id: 24
pumpernickel's OK-by worker_id: 15
lamination's OK-by worker_id: 7
leo@leo-VirtualBox:~/Documents/Project3/cis3207Project3/Project3$ wc -l logfile.
txt
400 logfile.txt
leo@leo-VirtualBox:~/Documents/Project3/cis3207Project3/Project3$
eadFromSocket: Please spell your word

eadFromSocket: lamination's OK
losing client Socket !

]-  Done                    ./clientdemo
]+  Done                    ./clientdemo
eo@leo-VirtualBox:~/Documents/Project3$
```



```
word: lamination's

readFromSocket: Please spell your word

readFromSocket: lamination's OK
closing client Socket !

[1]-  Done                    ./clientdemo
[2]+  Done                    ./clientdemo
leo@leo-VirtualBox:~/Documents/Project3$ telnet localhost 1024
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Please spell your word
leo
leo MISSPELLEDConnection closed by foreign host.
leo@leo-VirtualBox:~/Documents/Project3$ telnet localhost 1024
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Please spell your word
alabama
alabama OKConnection closed by foreign host.
leo@leo-VirtualBox:~/Documents/Project3$
```

All additional unit tests and files are located inside of test folder.
Additional omitted files may be found at github.com/lvergne1/cis3207Project3