Preliminary notes:
Once I made the proto file and set everything up correctly, I ran the command below to get the generated go files to pop up in the same directory as my primes.go file:

```
protoc --proto_path=. --go_out=. --go-grpc_out=. --go_opt=paths=source_relative --go-grpc_opt=paths=source_relative primes.proto
```

Compiled the proto file using that code above.

To run this system, you will have to open each of the three folders (main, workers, fileserver) and run the command "go build ." This will give you the executables. Now, to get each individual piece running you simply use "go run ." But more specific commands will be shown for the questions below.

Compiled and run the code for the fileserver, workers, and main go files by running that command above from the respective terminals associated with the folders. Start the main file, then the fileserver, then the workers.

1GB file: 26,368,536 primes
64MB file: 6,253,739 primes
32MB file: 1,599,522 primes

Note: I couldn't get the bash script working, so I just wanted to clarify that, yes, I ran straight from the terminal. So, I opened 4, 8 and 16 terminals. Also, because I got this project working so late, some of the statistics I couldn't complete. I just wanted to leave this note because I was still able to show proof that my system worked. Also, stuff like the statistics of jobs per worker were not considered b/c I just wanted to get this completed before time ran out. I could on

Project Report Questions:

1. What is the least (expected) elapsed (wall) time for a random datafile of 1GB in your implementation with M workers?

   RUN THE CODE IN THIS ORDER!!!
   Generate 1GB datafile: `head -c 1073741824 </dev/urandom >numbers.dat`
   Worker terminal code: `go run . -C 8192`
   Main terminal code: `go run . -N 67108864`
   Fileserver code: `go run .`

   Total Primes: 26,368,536
   AFTER ALL PRIMES ARE CALCULATED GO TO EACH TERMINAL & CONTROL - C!!!!

   `NOTE: if it took 11 minutes with 16 workers it'll take more with less workers`

Workers (M)	4	8	16
Elapsed Times	--- --- --- --- --- ---	--- --- --- --- --- ---	11 min 8.4 secs

2. What is the largest (random) datafile you can process within 3 mins (wall) elapsed time with M workers?

   RUN THE CODE IN THIS ORDER!!!
   Generate 64MB datafile: `head -c 67108864 </dev/urandom >numbers.dat`
   Worker terminal code: `go run . -C 8192`
   Main terminal code: `go run . -N 1048576`
   Fileserver terminal code: `go run .`

AFTER ALL PRIMES ARE CALCULATED GO TO EACH TERMINAL & CONTROL - C!!!!

Workers (M)	4	8	16
File Size (GB)	64MB	64MB	64 MB
Time	182000 msecs	118000 msecs	110000 msecs
Total Primes	--------------------- >	-------------------- >	6,253,739

3. *How does the elapsed time change as a function of M, for the datafile in Q2 and the N, C parameter values?*

I'm running out of time, and I have another assignment due at midnight, so I had to limit the calculations for this problem. Again, I hope this is enough to show proof it works so I can still get credit. I only had enough time to do this question for 4 and 8 workers. And I had to use an 4MB datafile which contains 393380 primes. Values taken for N and C were {64KB, 256KB} and {1KB, 4KB} respectively.

RUN CODE IN THIS ORDER!!

Worker terminal code: `go run . -C <value of C>`
Main terminal code: `go run . -N <value of N>`
Fileserver terminal code: `go run .`

AFTER ALL PRIMES ARE CALCULATED GO TO EACH TERMINAL & CONTROL - C!!!!

Values of bytes for ease of running commands:

1KB	**4KB**	**64KB**	**256KB**
1024	4096	65536	262144

N / C	**4 Workers**	**8 Workers**
64KB // 1KB	13278 msecs	9765 msecs
64KB // 4KB	13255 msecs	8281 msecs
256KB // 1KB	7536 msecs	3704 msecs
256KB // 4KB	8365 msecs	4339 msecs