TextProvider

For this assignment you'll create a system that serves lines from a file out to network clients. You can complete this assignment in any language, although one from Java, Python or C++/C is strongly preferred.

You are free to make use of any references or resources and any open source software that you find helpful, as long as you document their use. This is an individual assignment so please do not collaborate with others.

Specification

The TextProvider should serve individual lines from a static text file to clients over the network. The client-server protocol for this system should be the following:

```
GET <n>  => If <n> is a valid line number for the
         text file, you should return "OK\r\n" followed by the <n>th
         line from the text file.

         If <n> is NOT a valid line number, you should return
         "ERR\r\n".

         Note that the lines in the file should be indexed starting from 1,
         not 0.

QUIT     => This command should disconnect the client.

SHUTDOWN => This command should shutdown the server.
```

The TextProvider server:

```
* must support at least one client and may optionally support many concurrent clients
.
* must listen for client connections on TCP port 10322.
```

You can make the following assumptions about the format of the text file:

```
* Every line is newline ('\n') terminated
* Each line will fit into memory
* Every character in the file is valid ASCII; there will be no Unicode characters
```

You may perform any type of pre-processing on the file as long as the behavior of the server remains correct.

Your system should work well for both large and small files and it should work well as the number of GET requests/second increases.

Example client server interaction

Given a file whose contents are:

```
Humpty Dumpty sat on a wall,
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.
```

You might expect the following:

```
CLIENT -> GET 1
SERVER <- OK
SERVER <- Humpty Dumpty sat on a wall,
CLIENT -> GET -2
SERVER <- ERR
CLIENT -> GET 3
SERVER <- OK
SERVER <- All the king's horses and all the king's men
CLIENT -> QUIT
```

Environment

You can assume that your system will execute in an environment like that of a m4.xlarge EC2 instance running Cent OS 7.x. An m4.xlarge instance has the following properties:

```
* four vCPUs
* 16 GB of RAM
* EBS Storage
```

Nothing else will run on that machine so all the resources are available to your system.

Submission instructions

In the root directory of your solution you should include any documentation you write, scripts to run and build your system and a folder containing the source code.

- build.sh -> Builds your system so that it is ready to be used. You may shell out to a third party build tool like Ant, GNU make, Maven or Gradle. This script should download any libraries or other dependencies required by your program.
- run.sh -> Starts the server; accepts one command line argument which is the path to the file that the server will serve.
- README -> Should include the following:
    - A short walkthrough of the system's architecture (only needed if not included in the source as comments).
    - An explanation of the performance of your system with 1GB, 10GB and 100GB input files.
    - A discussion of what will happen if the # of client requests/second increases.
    - A list of external resources (blogs, websites, papers) you read when coming up with a solution to this exercise.
    - A list of libs or other third party tools used by the system.
    - Time spent on this assignment.

When you are satisfied with your solution, please create a .tar.gz and email back to us. We kindly ask that you NOT publish your solution to the Internet for others to find.