Testing,

Test the request, PUT, GET HEAD, the check if the program output the right response

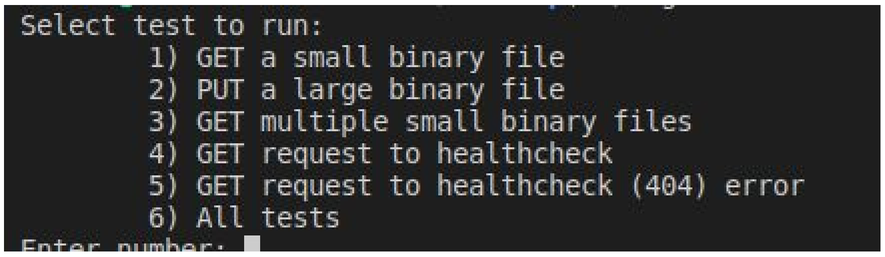
Test the logging,

Give some random request, check whether the content of the log file corresponds with the request.

Test the healcheck

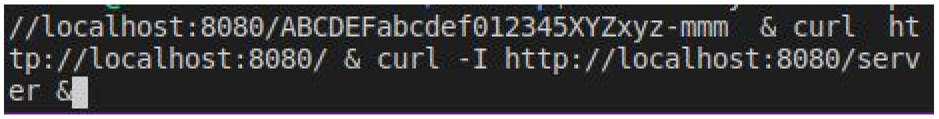
Give some random request both invalide and valid, check whether the healthcheck gives the right number of total processes and error.

Using the test script provided by piazza to check the program



```
Select test to run:
        1) GET a small binary file
        2) PUT a large binary file
        3) GET multiple small binary files
        4) GET request to healthcheck
        5) GET request to healthcheck (404) error
        6) All tests
Enter number:
```

Give multiple request to the client, test the functionality of multi thread



```
//localhost:8080/ABCDEFabcdef012345XYZxyz-mmm  & curl  ht
tp://localhost:8080/ & curl -I http://localhost:8080/serv
er &
```

Question

1 The multi threaded server is a little faster than the single thread server. For a single thread, it takes more than 1 minute to get a 400 MiB long text file. For multiple requests of about 400 MiB text file, the multithread are several seconds faster than the single thread.

2, the bottleneck will be mutex lock, since we enter the critical section, preventing other threads from proceeding. Also the writing into the log file is my bottleneck, since all the threads are waiting to write to the same file.writing all to the same file, they have to "queue" in order to get the write lock to the file, and this essentially sequentializes and slows the program down significantly.
In the worker part, there is full concurrency except, we use mutex lock when I get the client, and update the offset of byte for the logfile. For logging, there is less concurrency since since all the threads are waiting to write to the same file.writing all to the same file, they have to "queue" in order to get the write lock to the file, this is sequenialitzesd.

3. In the real life, the logging take much time than we can afford, all the threads are waiting to write to the same file.writing all to the same file, they have to "queue" in order to get the write lock to the file, and this essentially sequentializes and slows the program down significantly; the program take a few second to handle request, but it may takes a few minutes for logging the entire contents of files.