Assignment 3: Adding caching to httpserver
CSE130-01 Fall 2019
Due: Thursday, December 5 at midnight

Goals

The goal for Assignment 3 is to modify the HTTP server that you already implemented to have one additional feature: caching. Caching means that you are going to maintain a buffer in your server that contains a subset of the pages. When a request is received, if the requested page is in the cache, then it is read from the cache (if it is a GET request) or updated in the cache (if it is a PUT request). Otherwise, the page is first read from disk into the cache. In the log record of each request, you should indicate whether the page was in the cache at the time the request was received.

As usual, you must have source code, a design document and writeup along with your README.md in your git repository. Your code must build httpserver using make.

Design document

Before writing code for this assignment, as with every other assignment, you must write up a design document. Your design document must be called DESIGN.pdf, and must be in PDF. You can easily convert other document formats, including plain text, to PDF. Scanned-in design documents are fine, as long as they're legible and they reflect the code you actually wrote.

Your design document should describe the design of your code in enough detail that a knowledgeable programmer could duplicate your work. This includes descriptions of the data structures you use, non-trivial algorithms and formulas, and a description of each function with its purpose, inputs, outputs, and assumptions it makes about inputs or outputs

Since a lot of the system in Assignment 3 is similar to Assignments 1 and 2, we expect you're going to "copy" a good part of your design from your Assignments 1 and 2 design. This is fine, as long as it's your Assignments 1 and 2 you're copying from. This will let you focus on the new stuff in Assignment 3.

Program functionality

Your code may be either C or C++, but all source files must have a .cpp suffix and be compiled by clang++ version 7. As before, you may not use standard libraries for HTTP, nor any FILE * or iostream calls except for printing to the screen (e.g., error messages). You may use standard networking (and file system) system calls.

Your code for this assignment must be developed in asgn3.

Caching

Choose a caching page-replacement method from what we covered in the lectures (e.g., FIFO, NRU, Clock, and others) to implement it in your httpserver. The cache is going to be initially empty. When requests are received, if the requested page is not in the cache, then you bring it to cache from disk and place it in the cache. The cache has a size limit. The cache only maintains 4 files (do not worry about the file sizes, your cache's size limit is four files regardless of their size.) If you are bringing a page from disk to cache and the cache is full (hence, it has 4

files) then you must decide what page to replace using the page replacement algorithm you chose. When a page is replaced you must make sure to write it back if needed, i.e., if the page is dirty.

Caching can be turned on and off by a flag in your httpserver program (similar to how we used a flag to turn logging on and off.) The flag is -c.

Logging

If logging is active, then you should indicate in your log records whether the page was in cache when the request is received. This should be done by writing "[was in cache]" at the end of the first line if the page was in the cache and "[was not in cache]" otherwise.

For example:

```
PUT abcdefghij0123456789abcdefg length 36 [was not in cache]
00000000 65 68 6c 6c 3b 6f 68 20 6c 65 6f 6c 61 20 61 67 6e 69 20 3b
00000020 65 68 6c 6c 20 6f 54 28 65 68 43 20 72 61 29 73
========
GET abcdefghij0123456789abcdefg length 0 [was in cache]
========
```

README and Writeup

Your repository must also include a README file (README.md) and writeup (WRITEUP.pdf). The README may be in either plain text or have MarkDown annotations for things like bold, italics, and section headers. The file must always be called README.md; plaintext will look “normal” if considered as a MarkDown document. You can find more information about Markdown at https://www.markdownguide.org.

The README.md file should be short, and contain any instructions necessary for running your code. You should also list limitations or issues in README.md, telling a user if there are any known issues with your code.

Your WRITEUP.pdf is where you’ll describe the testing you did on your program and answer any short questions the assignment might ask. The testing can be unit testing (testing of individual functions or smaller pieces of the program) or whole-system testing, which involves running your code in particular scenarios.

For this assignment, please answer the following:

- Using your new httpserver with caching, perform an experiment to demonstrate how caching can improve performance (latency and/or throughput). Do a test with caching turned on and compare it with the same test but with caching turned off.

Submitting your assignment

All of your files for Assignment 3 must be in the asgn3 directory in your git repository. You should make sure that:

- There are no “bad” files in the asgn3 directory (i.e., object files).
- Your assignment builds in asgn3 using make to produce httpserver.
- All required files (DESIGN.pdf, README.md, WRITEUP.pdf) are present in asgn3.

You must submit the commit ID to canvas **before the deadline.**

Hints

• Start early on the design. This program builds on Assignments 1 and 2.

• Reuse your code from Assignments 1 and 2. No need to cite this; we expect you to do so.

• Go to section for additional help with the program. This is especially the case if you don’t understand something in this assignment!

• Aggressively check for and report errors.

• Use getopt(3) to parse options from the commandline. Read the man pages and see examples on how it’s used. Ask the course staff if you have difficulty using it after reading this material.

Grading

As with all of the assignments in this class, we will be grading you on all of the material you turn in, with the approximate distribution of points as follows: design document (35%); coding practices (15%); functionality (40%); writeup (10%).

If the httpserver does not compile, we cannot grade your assignment and you will receive no more than 5% of the grade.