Assignment 3: Back-up and Recovery in the HTTP server
CSE130-01 Fall 2020
Due: Thursday, December 10 at midnight

Updates and corrections:
- The GET requests will have a slash in front of the special file name. For example, GET /b and GET /r
- The timestamp format is the Unix timestamp which is the number of seconds since Jan 01 1970. (UTC)
- You can assume that we won't read or write to *.c and *.o files.

Goals

The goals for Assignment 3 are to modify the HTTP server that you already implemented to have two additional features: backups and recovery. Your HTTP server will have the ability to store a backup of all the files in the server and the ability to recover to an earlier backup.

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

We are not going to test multi-threading or redundancy in this assignment, so you can base of your code from asgn1 if you wanted.

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 Assignment 1 (and 2), we expect you're going to "copy" a good part of your design from your Assignment 1 (and 2) design. This is fine, as long as it's your Assignment 1 (or 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. 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.

We expect that you'll build on your server code from Assignment 1 (or 2) for Assignment 3. Remember, however, that your code for this assignment must be developed in folder asgn3, so copy it there before you start, and make sure you add it to your repository using git add.

Backups

A backup is a copy of the HTTP server data, which is all the files that can be accessed with GET requests. For example, if your HTTP server executable is in a folder, all the files in that folder are accessible to the HTTP server. The backup is a copy of all these files. To create a backup, your HTTP server would listen for special requests. The special request to create a backup is a request to GET a file named b. (you can assume that no file b would be in the folder and that b is reserved for this special backup request.) When the HTTP server receives a GET /b request, then it creates a new folder called "./backup-[timestamp]" where [timestamp] is the timestamp when the backup was created. Inside this new backup folder will be a copy of all the files (not folders) in the current folder (where the httpserver executable is).

Note: there will be no brackets in the timestamp

Recovery

The HTTP server has another special request for file r. When a GET /r is called, then the HTTP server restores the more recent backup. This means that all the files from the most recent backup folder will be copied to the current folder where the httpserver executable is.

Another request is Get /r/[timestamp] where the httpserver will restore the backup with the provided timestamp. If no backup with that timestamp exists, then an error 404 is returned.

List

The last special request is for file l. When a Get /l is called (note this is the small case of L), then the HTTP server will return a list of timestamps of all available backups. There will be a newline '\n' after each timestamp.

TESTING

Your design document is also 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. In particular, we want you to describe testing that you did for:

- Creating multiple backups and recovery to the most recent one or an earlier one.

QUESTION

For this assignment, please answer the following question in the design document:

- How would this backup/recovery functionality be useful in real-world scenarios?

README

Your repository must also include a README file (README.md). 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.

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) are present in asgn3.
- Any scripts or files you used for testing (optional)

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

Hints

• Start early on the design. This program builds on Assignment 1 (and 2). If you didn’t get Assignment 1 (and 2) to work, please see the course staff ASAP for help getting it to work.

• Reuse your code from Assignment 1 (or 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.

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 (40%); coding practices (10%); functionality (50%).

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