Static Web Server

Description

This lab aims to practice implementing a web server that hosts static web content. Please follow the instructions and have fun!

1. Please download the homework development and testing package (or HW package for short) from here. With the HW package, you should be able to implement your homework offline.

   The HW package should be compatible with the following runtime:

   - Docker in native Linux, Mac OS, and WSL. If you use M1/M2, ensure you have Rosetta installed.
   - Docker in the classroom virtual machine

   We have updated the homework package to support:

   - A builder docker instance.
   - Revised sample codes to read port numbers and document root path from command line.

   Additional notes will be highlighted in red texts or red blocks.

2. To play with the HW package, unpack the files and run docker-compose up -d in the unpacked hw_staticweb directory. The command creates three containers.

   - **sw_lighttpd**, a minimal lighttpd installation that is for you to inspect how a web server works.
   - **sw_demo**, a dummy web server implementation that does nothing but simply close a connected client connection, and
   - **sw_tester**, a docker used for testing your implementation.
   - **sw_builder**, *a docker used for building your source codes. This would be useful if you run everything from native Mac OS or Windows terminal instead of a Linux OS.*

   If you want to rebuild all the dockers, e.g., rebuild from an updated package, you may use the following commands:

   docker-compose stop # stop all containers
   docker-compose rm # remove all containers
   docker-compose up -d --build

3. The **sw_lighttpd** container exports HTTP and HTTPS servers on ports 10801 and 10841. You can access the service using command lines such as

   curl http://localhost:10801/
   curl -k https://localhost:10841/

   You can also visit the URLs using your browser. Alternatively, here are two other URLs for demo purposes http-based and https-based.

Because the server uses a self-signed certificate, you may receive warning messages on connecting to the server.

4. You must implement your static web server in the demo directory. You can modify anything in the demo directory, but please ensure that you can produce a *statically linked* executable named server in the directory (using make command).

   We demonstrate the compilation process on a native Linux OS with docker runtime. In that case, you can use the make commands in the console. However, if you work with native Windows or Mac OS console, you may use make builder to compile files in **./demo** directory.
5. The **sw_demo** container invokes the server executable placed in the demo container. Similar to the **sw_lighttpd** container, it also exports two ports, 10802 and 10842, for accepting HTTP and HTTPS services, respectively. The container may keep restarting if the server executable cannot be invoked appropriately.
6. To test your implementation, run the command

   docker exec -it sw_tester /testcase/run.sh lighttpd

   # - or -

   docker exec -it sw_tester /testcase/run.sh demo

   The last parameter determines whether you run the test cases against the ***lighttpd*** implementation or *your **demo** implementation*.
7. For the expected behavior of your implementation, please read the specification section for more details.

Specification

1. You are requested to implement parts of the HTTP/1.0 specification defined in RFC1945.
2. You only need to serve static files from the/html directory in the container. In case your server receives additional parameters in a request, e.g., a query string, your server should ignore that.
3. You only need to handle GET requests and ensure that a modern browser, such as Chrome, can browse files using your implmentation. The demo link to your server is available here (http and https).
4. Your server should respond appropriate MIME types for files based on a file's extension name. A list of typical MIME types is available here. You may only implement that for text, image, and audio files.
5. The default index file for a directory is index.html. If there is no index.html file in the directory, your server should return 403 status code.
6. On accessing a non-existent file or directory, your server should return a 404 status code.

7. For requesting a directory without a trailing slash, your server should return a 301 status code and redirect it to the one with a trailing slash.
8. For unsupported HTTP request methods, your server should return a 501 status code.
9. The sw_tester container performs workload testing against your server. Please ensure your implementation is efficient and reliable.
10. One optional requirement is to implement the support of HTTPS connections. You may leverage any SSL/TLS library you like. For example, bearssl, mbedtls, openssl, or gnutls. In case you need the server certificate and key, they are available in the /cert directory in the container.

Demonstration

1. [10pts] Your server passes all simple GET requests.
2. [10pts] Your server returns status code 301 as expected.
3. [10pts] Your server returns status code 403 as expected.
4. [10pts] Your server returns status code 404 as expected.
5. [10pts] Your server returns status code 501 as expected.
6. [10pts] Your server passes the hidden test cases from the TAs.
   Please download the compressed file, unzip it, and perform the following two steps:
   i. Move the entire “hidden” folder to the “data” folder
   ii. Move the file “run_hidden.sh” to the “testcase” folder
   Afterward, you can use “run_hidden.sh” to run the hidden test cases.
   If facing permission error, please follow the following steps:
   iii. Remove **ro** in **line 43** and **line 44** in docker-compose.yml
   iv. run the commands below:
   dokcer-compose build --no-cache docker-compose up -d docker exec -it sw_tester ash cd testcase/ chmod +x ./run-hidden.sh
   samples:

```
tester:
container_name: sw_tester
build: .
image: chuang/staticweb
restart: unless-stopped
volumes:
  - ./cert:/cert:ro
  - ./data:/html
  - ./testcase:/testcase
networks:
```

default:

If facing Chinese encoding problem for our Chinese case, please help us to replace the old folder name（一堆亂碼） to the correct Chinese folder name(中文資料夾).

7. [20pts] All resources listed in the demo.html page work in a typical browser.
8. [20pts] Your server passes the workload test. The score will be given based on the ranking of your implementation among all students in the class. There will be only four ranks. The top-ranked, second-ranked, third-ranked, and bottom-ranked implementations get 20pts, 15pts, 10pts, and 5pts, respectively.
9. [20pts] If your server supports SSL/TLS and passes most of the test cases listed above.