CSE-381: Systems 2

Homework #5

Due: Wed March 10 2021 before 11:59 PM

Email-based help Cutoff: 5:00 PM on Tue, March 9 2021

Maximum Points for: 5 + 7 + 3 + 3 = 18 points

Objective

The objective of this part of the homework is to develop 1 C++ program to:

- Develop a custom web-server how the `bash` shell runs commands
- Recollect the use of HTTP protocol for communication from prior projects
- Develop a web-server that can process HTTP GET requests
  - Chunk data from a given file
  - Run a program with command-line arguments and return its outputs back to the client
  - Work with anonymous pipes
- Continue to gain familiarity with I/O streams & string processing
- **Learn functional testing of programs**
- **Do not use global variables**

Submission Instructions

This homework assignment must be turned-in electronically via the CODE Canvas plug-in. First ensure your program compiles without any warnings or style violations. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload just your updated source file (named with convention `MUID_hw5.cpp`) and screenshot as required.

General Note: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Grading Rubric:

The programs submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet base case requirements will be assigned zero score!
Program that do not compile, have a method longer than 25 lines or badly formatted code as in } } }, etc., or just some skeleton code will be assigned zero score.

- Point distribution for various commands:
  - Base case: Process HTTP request and send file contents as chunked HTTP response [**5 points**]
  - Additional feature: Process HTTP request, run a given command, and send output of the command as chunked HTTP response [**7 points**]
    - Browser screenshot: A screenshot showing full window of a browser (along with location bar) showing correct operation of the additional feature from a web-browser. [**3 points**]

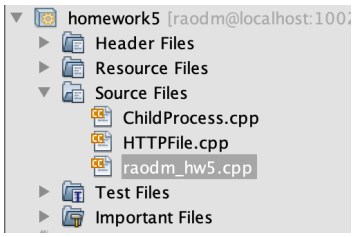
  - Formatting, Good organization, code reuse, documentation, etc. – **earned fully only if base & additional functionality operate correctly**: [1 + 2 = **3 points**].
- **-1 Points**: for each warning generated by the compiler when using Miami University C++ project (warnings are most likely sources of errors in C++ programs)
- **-18 Points**: If global variables (without namespaces) are used.

Develop an amazing custom web-server

Background

In this homework you will be developing a simple program to process HTTP requests and generate responses. Recollect that HTTP is a multi-line text protocol that is used by World Wide Web (WWW), an application running on the Internet. In this project, you will be developing a program to respond to HTTP requests. **Overall, this program is just a simple extension of the previous lab. Most of the core functionality already provided by starter code or from previous exercise(s).**

Setup



You are given headers and source files that provide several functionalities. The `ChildProcess` class is exactly the same one developed in lab exercises. The `main` file is essentially a minor extension from Homework #1 (with `url_decode` method added). Nevertheless, review these files to form a mental model what you already have available. Then **write pseudocode to plan** out the operations you want to get done. Then go about implementing your solution. The setup of all of the files in NetBeans is show in the adjacent screenshot. Ensure all of the supplied files are copied to your NetBeans project folder.

Request Processing Requirements

You are required to implement the `serveClient` method in `main.cpp` file in the supplied starter code. This method must be implemented (using additional helper methods to be implemented by you) to process a HTTP `GET` request (1st line of a HTTP request) in the following manner. First read and discard HTTP headers (this is important because your solution will not work with browsers).

- **Base case – Return output of a given file [5 points]**: If the requested resource does not start with "`/cgi-bin/exec`" then assume this request is for a given file. Use the supplied `HTTPFile` class method to return the contents of a given file as shown in the example below. Ensure you url decode the path first (using supplied helper method).

```
os << http::file(path);
```

- **Additional case – Return output of a program [7 points]**: If the input GET line starts with "`/cgi-bin/exec?cmd=ls%20-l`" then run the command specified after the `cmd=` as in `ls -l` (in this example). Of course, use the supplied `ChildProcess` class which is the one you developed in prior exercise.

Functional testing

When testing your program during development, simply copy-paste the inputs to your program (mimicking as if you manually typed the inputs). Once you have done the basic testing you may redirect input and output for testing as shown below. Several test inputs and expected output files are also supplied to streamline your testing as shown below:

```
$ ./hw3 test_files/base_case1_inputs.txt > my_base_case1_output.txt
$ diff my_base_case1_output.txt test_files/base_case1_outputs.txt
```

Note: If your solution is correct, then the above `diff` command will not print any output.

Ensure you test with all input files and corresponding expected outputs to check all features of your program.

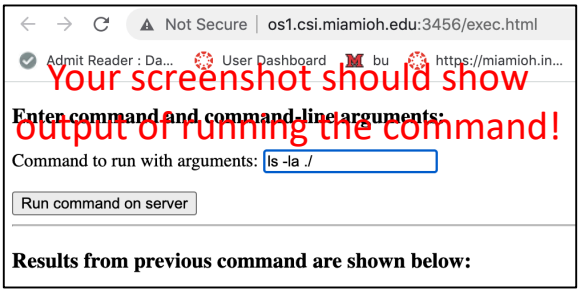
Testing via a browser

The starter code enables the program to run as a web-server on an available port number. The starter code prints the port number, say **3456**. Each time the program is run, the port number will change. Given a port number, say **3456**, you **must test the program via a web-server** by suitably changing the command and arguments in the following URL:

1. http://localhost:3456/test.txt
2. http://localhost:3456/cgi-bin/exec?cmd=ls -l
3. http://localhost:3456/cgi-bin/exec?cmd=echo "hello"

Browser testing [3 points]

you **must test the program via a web-server** by suitably changing the port number in the following URL: http://localhost:3456/exec.html. You should see a from as shown in the adjacent screenshot. Enter the command `ls -la ./` as shown in the screenshot and click the "Run command on server" button. You should see the output of the command on your browser window. Congratulations! Now, you have a fully operational web-server. This type of web-server is what is used to run programs in other languages like Java, php, php/Laravel, python, etc. Now make a screenshot of your entire browser window (clearly showing URL in location bar) and submit the image (saved as jpeg/jpg file) it along with your source code.



Submit to Canvas

This homework assignment must be turned-in electronically via CODE Canvas plug-in. Ensure all of your program(s) compile without any warnings or style violations and operate correctly. Once you have tested your implementation, upload:

1. Your updated C++ source file named with the convention `MUID_hw5.cpp` onto Canvas.
2. Upload screenshot of web-browser showing output of running specified command.

Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.