Homework 2 Report

HOW TO RUN AND TEST

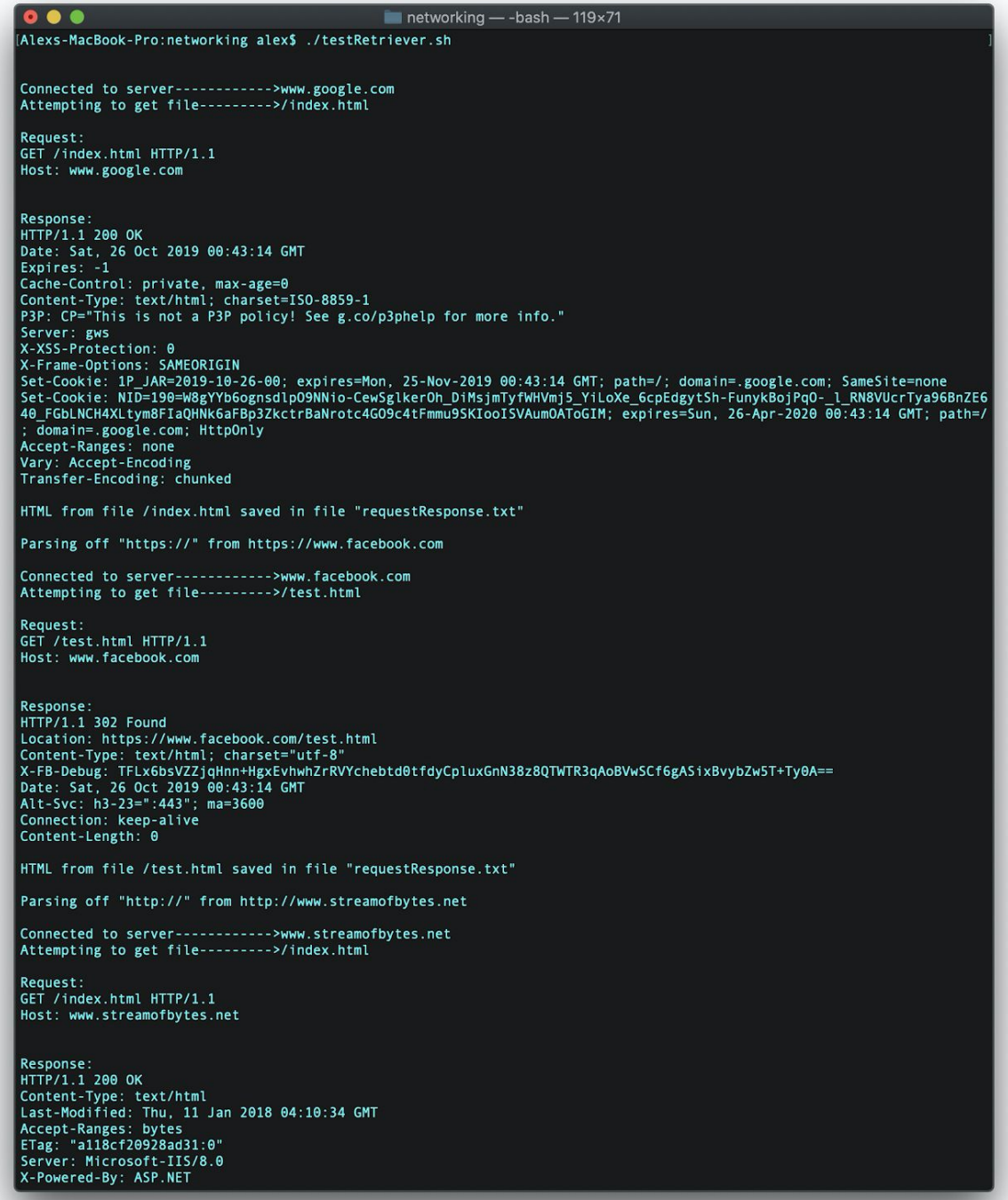
1. If you're reading this, you've already unzipped the file. Congrats, you're already on your way to testing my program and grading me.
2. I've included executable files already if you just want to run those. Who wants to do that though. I've included scripts, so you can just run those.
3. When in the directory of my files please run this code in the terminal:

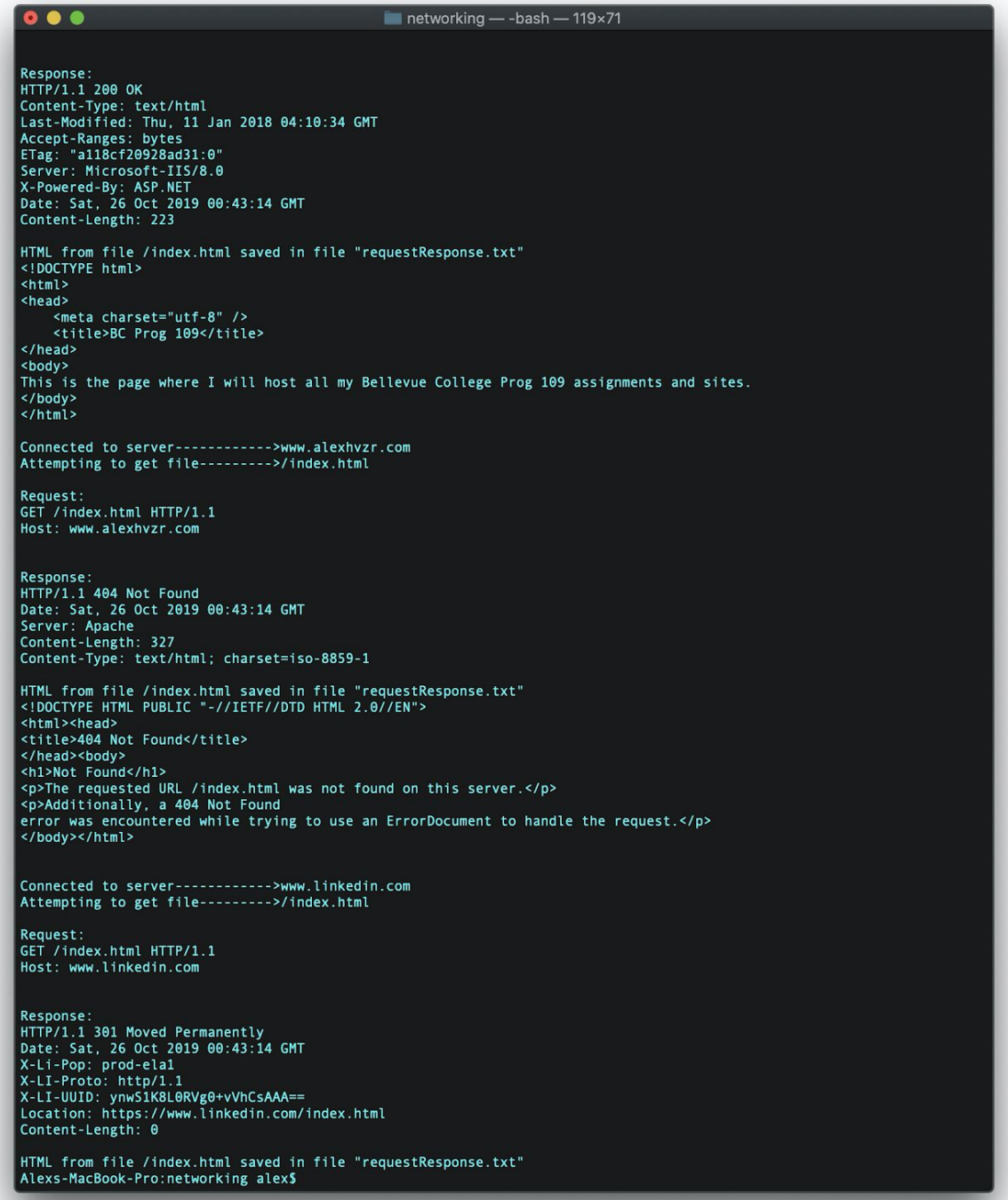
Side note: you may need to run chmod +x <file>.sh to run the scripts so you have permissions.

   a. ./removeExecutables.sh
   b. ./build.sh
      i. Build will make executable files for both the server and the retriever.
   c. chmod +x testRetriever.sh
      i. You're almost there!! Take a break and have a snack, we can continue when you're done.
   d. ./testRetriever.sh
      i. You've done it, look at that output!

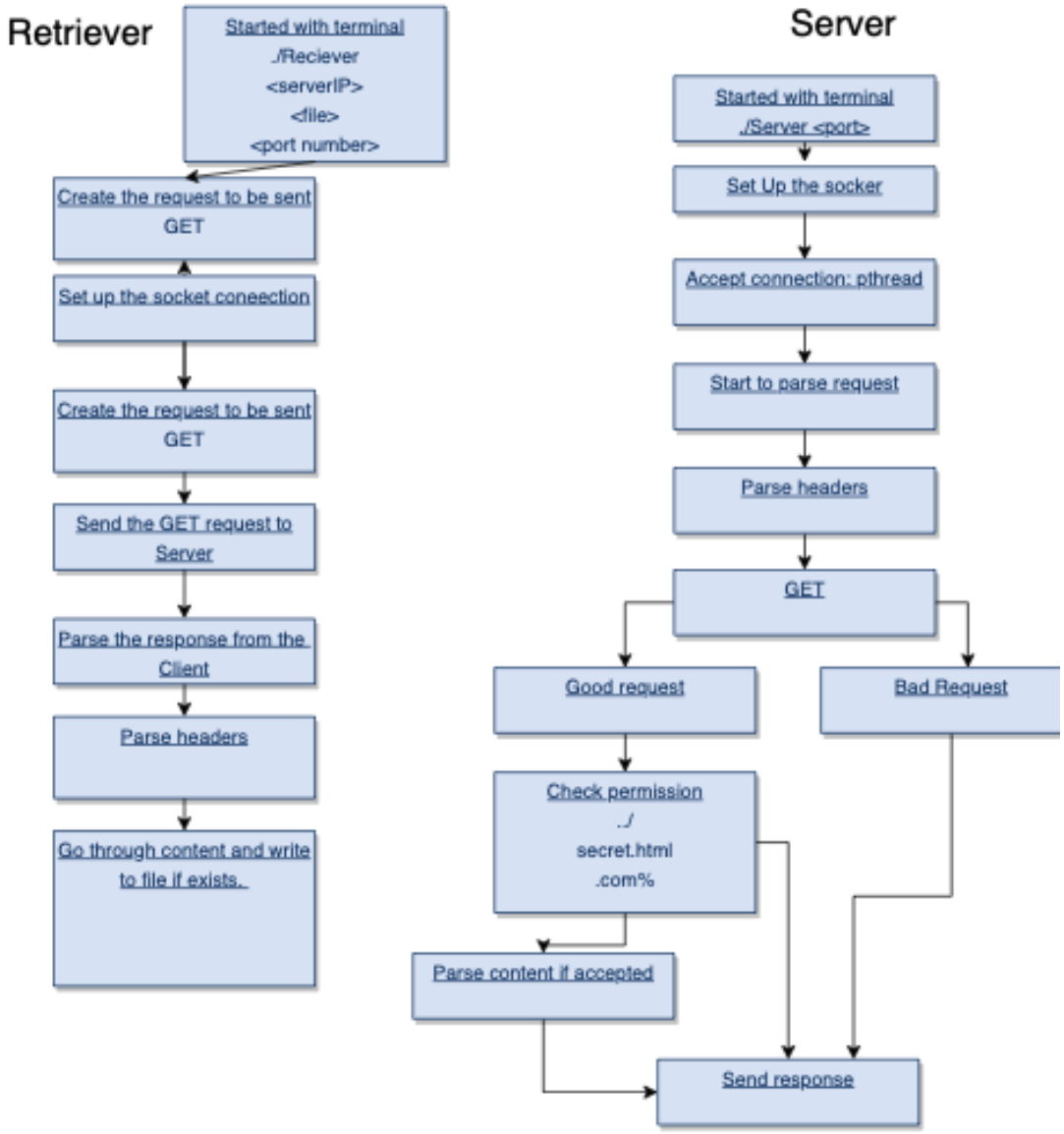
The file that the retriever saves to is cleared each time a test is ran and saves the output of the html at the specified file if there is one available. In the testRetriever.sh file there is cat requestResponse.txt which is where each separate response from the sites are saved.

This is what the output should look like when you have ran the tests.




networking — -bash — 119×71
Response:
HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Thu, 11 Jan 2018 04:10:34 GMT
Accept-Ranges: bytes
ETag: "a118cf20928ad31:0"
Server: Microsoft-IIS/8.0
X-Powered-By: ASP.NET
Date: Sat, 26 Oct 2019 00:43:14 GMT
Content-Length: 223
HTML from file /index.html saved in file "requestResponse.txt"
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>BC Prog 109</title>
</head>
<body>
This is the page where I will host all my Bellevue College Prog 109 assignments and sites.
</body>
</html>
Connected to server---------->www.alexhvzr.com
Attempting to get file--------->/index.html
Request:
GET /index.html HTTP/1.1
Host: www.alexhvzr.com
Response:
HTTP/1.1 404 Not Found
Date: Sat, 26 Oct 2019 00:43:14 GMT
Server: Apache
Content-Length: 327
Content-Type: text/html; charset=iso-8859-1
HTML from file /index.html saved in file "requestResponse.txt"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
The requested URL /index.html was not found on this server.
Additionally, a 404 Not Found
error was encountered while trying to use an ErrorDocument to handle the request.
</body></html>
Connected to server---------->www.linkedin.com
Attempting to get file--------->/index.html
Request:
GET /index.html HTTP/1.1
Host: www.linkedin.com
Response:
HTTP/1.1 301 Moved Permanently
Date: Sat, 26 Oct 2019 00:43:14 GMT
X-Li-Pop: prod-ela1
X-LI-Proto: http/1.1
X-LI-UUID: ynwS1K8L0RVg0+vVhCsAAA==
Location: https://www.linkedin.com/index.html
Content-Length: 0
HTML from file /index.html saved in file "requestResponse.txt"
Alexs-MacBook-Pro:networking alex$

The way the retriever and server works is illustrated by the flow diagram charts below:



This project was to create a simplified http request (only GET) in c++. The requirements were as listed:

1. Overall Requirement

You will write two programs that exercise a simplified version of HTTP. **The retriever will work in conjunction with any web server and the server will work in conjunction with any web browser**. This way, you can test your software independently of each other.

- You need to write C++ implement the retriever and server.
- You should not use existing libraries to retrieve HTTP files or perform socket communication.
- You need to write both the build script and demo script files using bash shell script.

2. Detailed Requirement

- Retriever
    - Your retriever takes in an input from the command line and parses the server address and file (web page) that is being requested.
    - The program then issues a GET request to the server for the requested file.
    - When the file is returned by the server, the retriever outputs the file to the screen and saves the retrieved file to the file system.
    - If the server returns an error code instead of an OK code, then the retriever should not save the file and should display on the screen whatever error page was sent with the error.
    - Your retriever should exit after receiving the response.
    - The server name can be an IP address for simplicity.
- Server
    - Your server waits for a connection and an HTTP GET request (Please perform multi-threaded handling).
    - Your server only needs to respond to HTTP GET request.
    - After receiving the GET request

            - If the file exists, the server opens the file that is requested and sends it (along with the HTTP 200 OK code, of course).
            - If the file requested does not exist, the server should return a 404 Not Found code along with a custom File Not Found page.
            - If the HTTP request is for SecretFile.html then the web server should return a 401 Unauthorized.
            - If the request is for a file that is above the directory where your web server is running (for example, "GET ../../../etc/passwd"), you should return a 403 Forbidden.
            - Finally, if your server cannot understand the request, return a 400 Bad Request.
        - After you handle the request, your server should return to waiting for the next request.
- Build script
    - You need to provide a build script to build your retriever and server.
- Test script
    - You need to create a script to test your retriever. This script should help you test your retriever when retrieving "authorized," "unauthorized," and "forbidden" files.

3. What to submit

You must submit the following deliverables in a zip file. If your code does not work in such a way that it could have possibly generated your results, you will not receive credit for your results. Some points in the documentation, evaluation, and discussion sections will be based on the overall professionalism of the document you turn in. You should make it look like something you are giving to your boss and not just a large block of unorganized text.

Your code should include both server and retriever, with a build and demo script. The demo script should run through all of the following test cases, together with screenshots of your programs executing all of the test cases. Make sure your screenshots are properly labeled!

1) Real Web browser accessing your server (screenshot only)

2) Your retriever accessing a real server (screenshot and demo script)

3) Your retriever accessing a file from your server (demo script)

4) Your retriever requesting an unauthorized file from your server (demo script)

5) Your retriever requesting a forbidden file from your server (demo script)

6) Your retriever requesting a non-existent file from your server (demo script)

7) Your retriever sending a malformed request to your server (demo script)