Sahil Gandhi and Arpit Jasapara

CS 118 / Winter 2018  UIDs: 704-598-105 and 504-742-401

CS 118 Project 1 report

1. **High-level Description of Server Design:**

Our server uses sockets to bind a connection with incoming client requests. When a client requests to connect with the server, it accepts the connection and spawns a child process for the client. The child process reads in the GET request, outputs it to console, parses the file name and file type, opens and reads in the file’s data and metadata using fstream and fstat, and appropriately formats it into an HTTP response. The HTTP response is then output to console and sent back to the client to be downloaded or displayed appropriately within the client’s browser.

1. **Difficulties that we faced and how we solved them:**

One major difficulty that we faced was how to open the file and get its data. We could use the C open method or use C++ fstream, since both have their advantages and disadvantages. We ultimately ended up using both, where we used the C open method in combination with fstat to quickly obtain the file’s metadata, and fstream to actually read in the file’s data efficiently.

1. **Manual to compile and run the source code:**

Run “make” to compile the program. Then run “./p1\_server” to have the server up and running. Attached are screenshots of how to do this:

Making program:



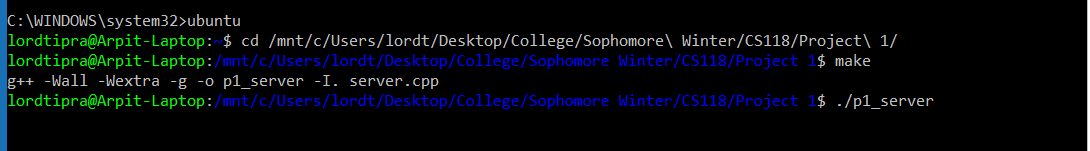
Running program:



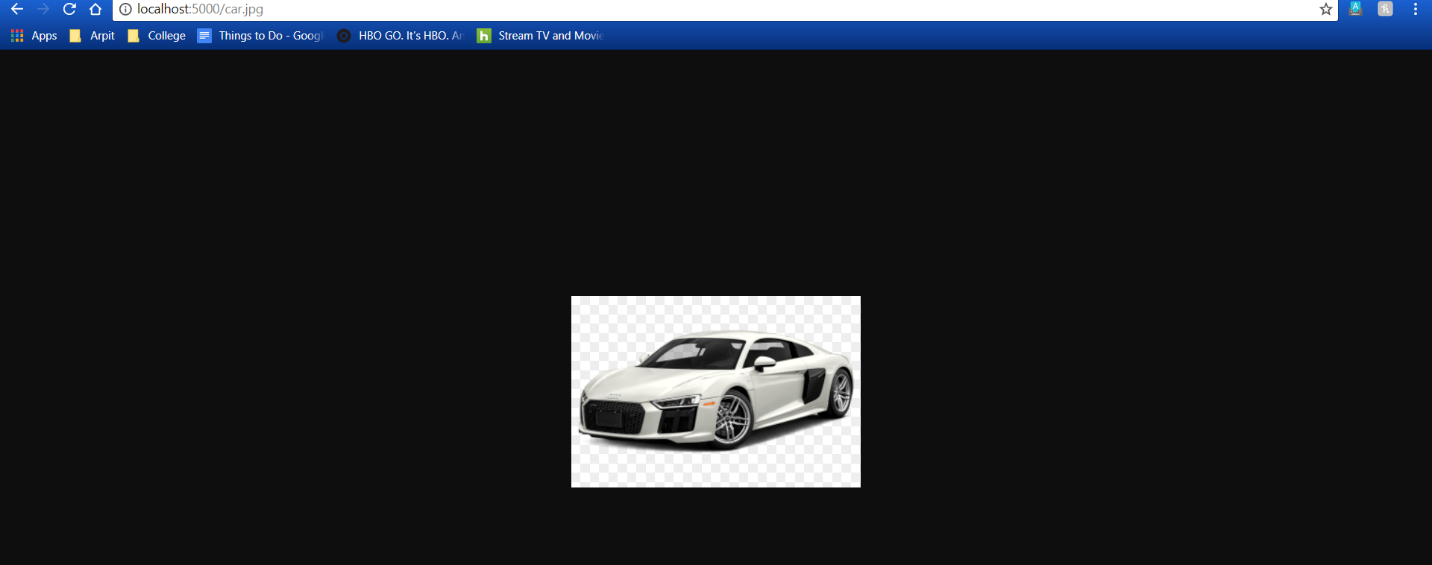
1. **Sample outputs of client-server (Part A and Part B)**
2. Part A sample response(s)



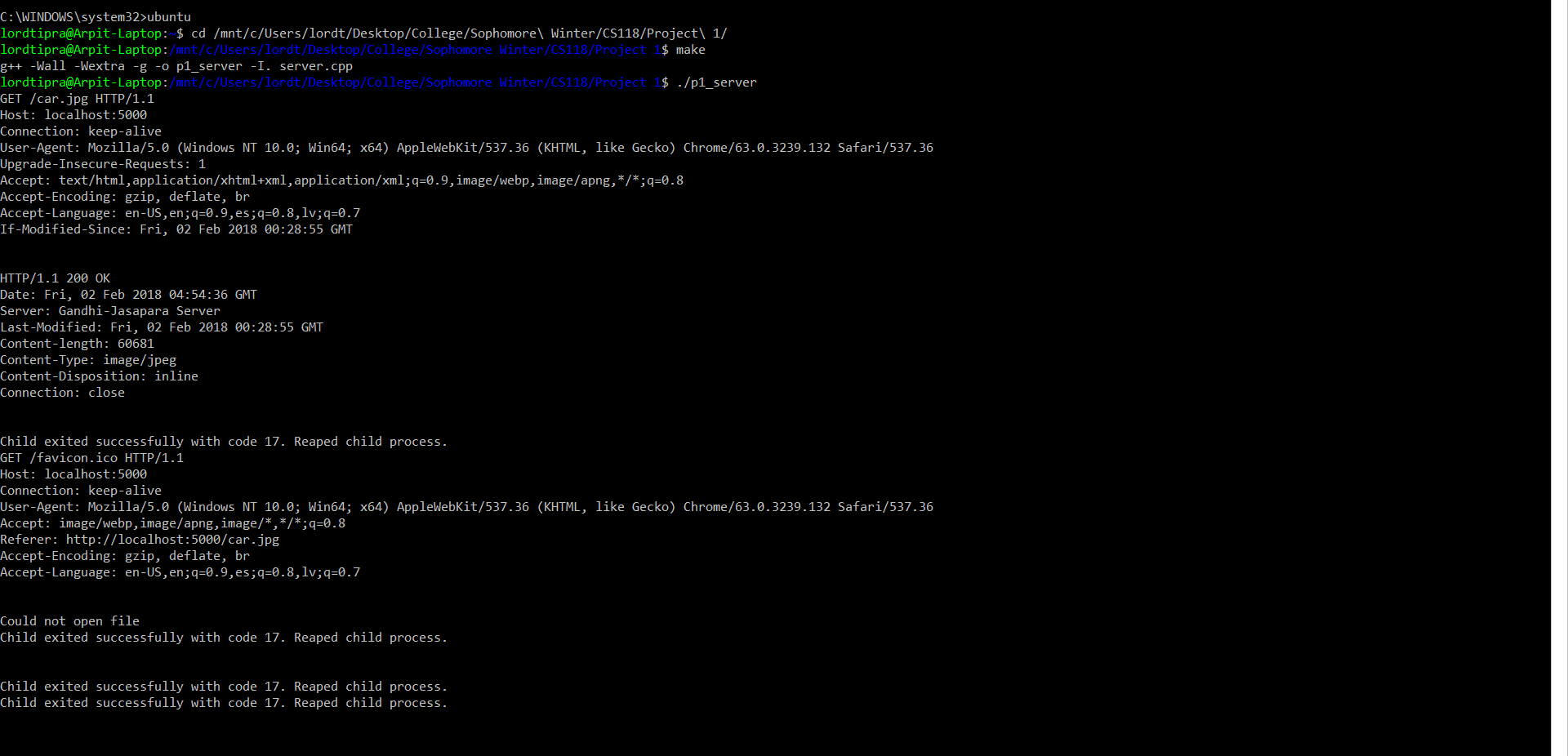
As we can see from this image, the server outputs the GET request from the browser successfully.

1. Part B sample response(s)

As we can see from this image, we made our project and ran the server.

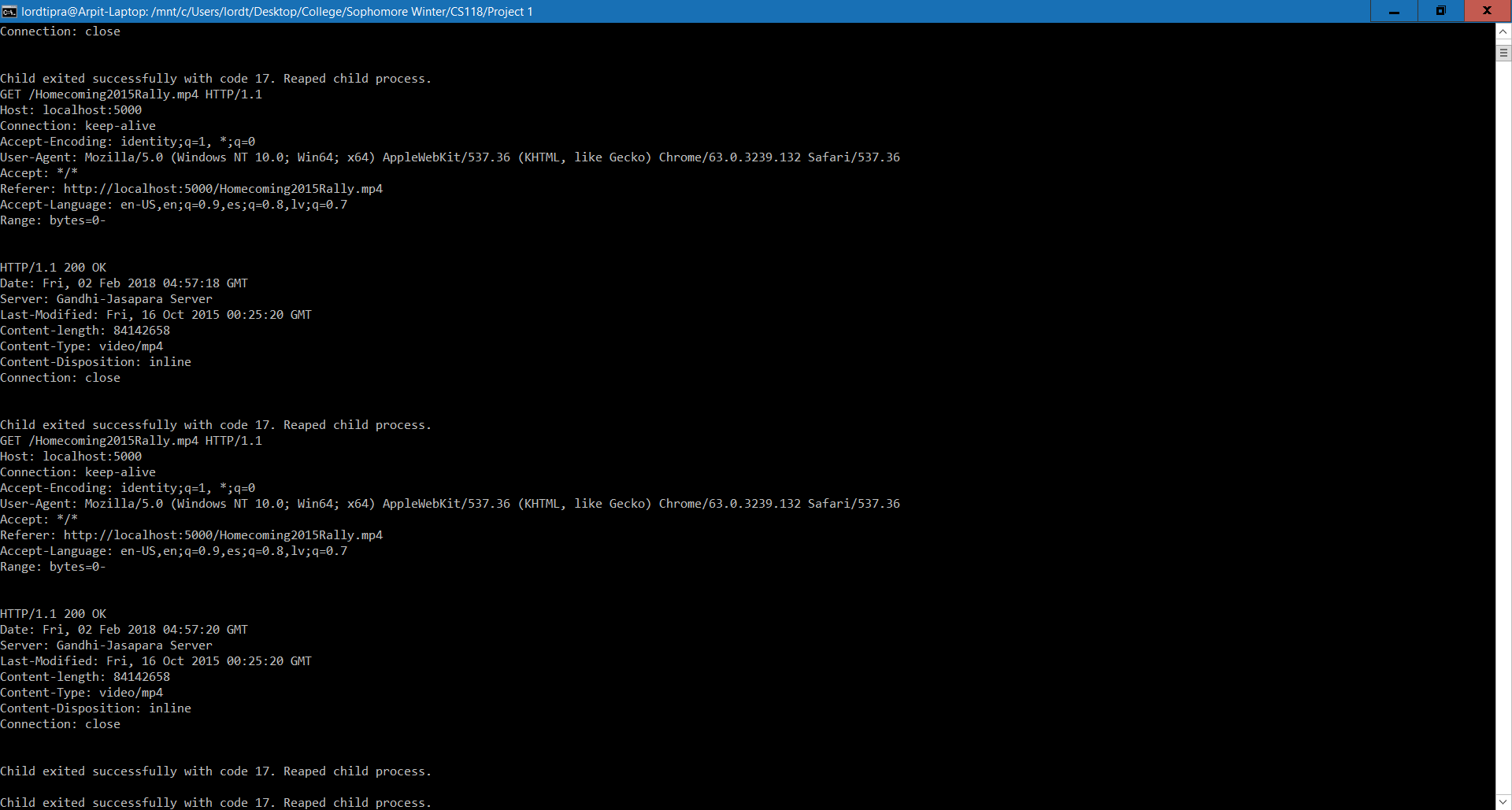


As we can see from this image, the GET request for a .jpg file (car.jpg) successfully shows up in the browser due to the HTTP response.



As we can see from this image, we successfully received the GET request for the car.jpg, processed it, and sent back an appropriate HTTP response.





As you can see from these images, we added functionality for .mp4 and several other file types (.pdf, .wmv, etc.) as part of the extra credit and to make our HTTP web server more robust. We successfully received the GET request and were able to display the video in-browser. Moreover, we sent back the appropriate HTTP response, and reaped the child process thereafter.