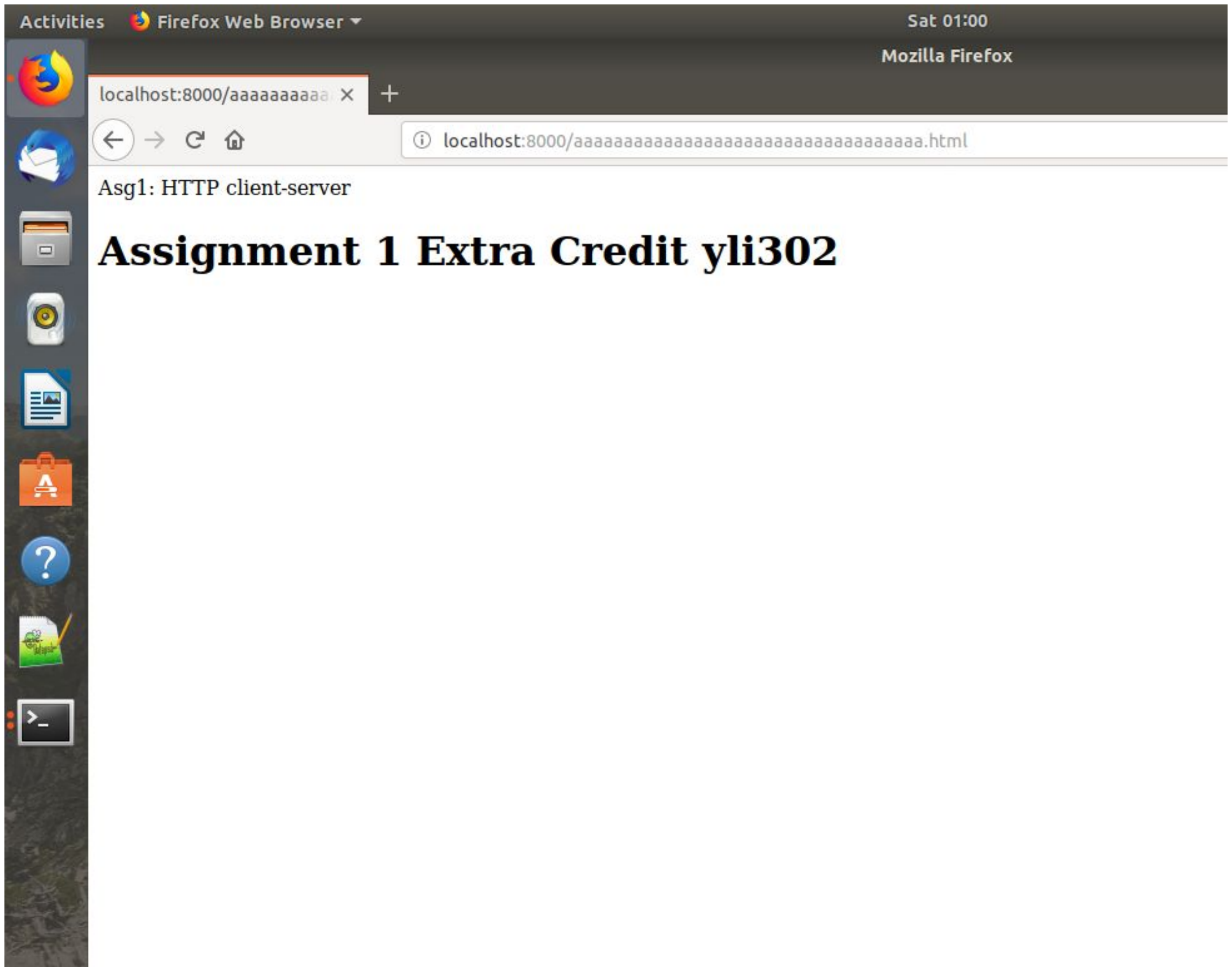
Write up of ASG0: mycat

Yujia Li
CruzID: yli302

Extra Credit:


Activities
Firefox Web Browser
Sat 01:00
Mozilla Firefox
localhost:8000/aaaaaaaaaa
localhost:8000/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.html

Asg1: HTTP client-server

Assignment 1 Extra Credit yli302

Write up questions:

• Briefly discuss (but do not implement) the changes you'd need to make to allow the client to send and receive from stdin and stdout, respectively.

For PUT request, I should: 1. read Data from local file descriptor
2. send a Request Header
3. send Data to server.
4. receive response from server.

If I want to send data from stdin, I need to change first step: let client read Data from stdin, instead of file descriptor.

And I also need to change my argument check, let argument like `s::httpname` or `s:stdin:httpname` be process by client.

For GET request, I should: 1. send a Request Header
2. receive response from server.
3. receive Data from server.
4. write data to local file descriptor

If I want to receive data to stdout, I need to change fourth step: let client write Data to stdout, instead of file descriptor.
And I also need to change my argument check, let argument like `r:httpname:` or `r:httpname:stdout` be process by client.

• **What would happen in your implementation if, during a transfer, the connection was closed, ending the communication early? This extra concern was not present in your implementation of mycat. Why not? Hint: this is an example of complexity being added by an extension of requirements (in this case, data transfer over a network). You do not need to test or implement this, just discuss it.**

I implementation the header with `Content-Length: number\r\n\r\n` in both PUT and GET. I will parse the header and get correct content length before receiving the Data. The receiver will close the connection in this condition:

```
if(recv return 0 || recv length == Content-Length)
```

So, if communication close early, the sender will not be affect, but the receiver will stop receive because recv return 0. And data received is less than Content-Length mentioned. So I will not recv all data.

In mycat, I can read until the end of file and write data, which I have read, to another file, and do not worry that read returns a 0 before end of file. In other words, I will always read the whole file, and write the whole file to another place.

•**Work with another student in the class and use your client to talk to their server, and their client to talk to your server. You should each run the client and server on your own VM. Report on your experiences, including the name and CruzID of the person you worked with. You may do this with more than one student if you like. REMEMBER: while we want you to try your server and their client (and vice versa), you may not show your design or code to another student. Just try client-server interoperation, and let the protocol do the talking.**

I tested my HTTP client-server with two person:
Adam Askari: haskari@ucsc.edu
Hang Yuan: hyuan3@ucsc.edu

Because I have implement Content-Length in both client and server, so many error appear when I first test. I change many design to handle implement without Content-Length.

Tests are in test part:

TEST:

1. My client with my server:
   S:filename:httpname
      Request PUT or GET(must include at least one request which will be responded other than 200 ok or 201 create) many times(at least 5 times) in very short time(1~2 seconds) in different ./httpclient process will lead to write incorrect data to file in PUT request.
   r:httpname:filename
      No known bug.
2. My Client:
   a. With python SimpleHTTPServer:
      s:filename:httpname
         Don’t support PUT request
      r:httpname:filename
         Work with SimpleHTTPServer, if delete the check of HTTP/1.1.(HTTP version of SimpleHTTPServer is HTTP/1.0)
   b. With Adam’s server:
      S:filename:httpname
         Sometimes, his server will show bad address error and response nothing, my client will wait for response, and will not end.
      r:httpname:filename
         No known bug.
   c. With Hang’s server:
      S:filename:httpname
         No known bug.
      r:httpname:filename
         No known bug.

3. My Server:
   a. With curl:
      No known bug.
   b. With Adam’s client:
      No known bug
   c. With Hang’s server:
      No known bug