COMS W3157 Advanced Programming, Lab #8 (0 points - optional lab)

Part 1: Build and install your own Apache web server

In this part, you will set up your own Apache HTTP server.

- First, extract the Apache HTTP Server source code (available at /home/jae/cs3157-pub/bin/httpd-2.2.8.tar.gz) into your lab8 directory. See the man page for tar command for how to do this.

- Build and install it. See INSTALL file for instruction.

  The install directory (which you pass "./configure" using "--prefix" option) MUST be "apache2" directory directly under lab8 directory.

- Change the following things in the server configuration file (httpd.conf):

  - Listening port number: pick a number that's unlikely to clash with others and tell us in your README.txt.

  - Make the necessary changes in httpd.conf so that the document root directory is /home/jae/html

  - Note that a copy of the original httpd.conf is in "original" directory. Use diff command to see what you changed.

You can start, stop, or restart your server by running apachectl in the apache2/bin directory:

```
apachectl start
apachectl stop
apachectl restart
```

Try hitting your Apache server with web browser. Access the Star Trek page. Apache server logs every request into "access_log" file. You can open another terminal and look at the file as it gets updated:

```
tail -f access_log
```

Also check out error_log in the same directory.

Make sure you stop the server with "apachectl stop" before you log out. Otherwise, your server will keep running on the machine even after you log out.

Include the following in your README.txt:

- The (part of) "ps ajxfww" output that shows your Apache server running.

- The output of diff command showing what you changed in httpd.conf.

- The capture of a netcat session fetching the Star Trek index.html page from your own Apache server.

Part 2: Build a simple module and load it up in your Apache

Learn how to write an Apache module by reading Chapter 5.1 of "The Apache Module Book" by Nick Kew, which is available via Safari books online, which you can access it for free from computers in the Columbia University network. (A PDF for the chapter is in /home/jae/cs3157-pub/bin as well.)

Follow the mod_helloworld.c example in Chapter 5.1.

Once you get it working, make the following additions:

(1) Display hit count.

Declare a file-scope static variable (at the top of the file, outside of any function) like this:

```
static int HitCount = 0;
```

And include it in the HTML page you generate using:

```
ap_rprintf(request_rec *, const char *fmt, ...);
```

which works similar to printf().

(2) Display "unparsed_uri", "uri", and "args" members of the request_rec structure, formatted in an HTML table.

Send the following query from your browser to see what values those 3 strings take:

```
/helloworld?key=abc
```

Include the following in your README.txt:

- The values of the 3 members of the request_rec structure when you sent the "/helloworld?key=abc" request.

- Answer to the following question (in your README.txt):

  You will see the hit count go up if you keep hitting refresh in your browser displaying the page generated by the helloworld module.

  However, as you keep hitting it (you can hold down Ctrl-R in FireFox), you will see a strange behavior. What do you see? Can you explain what's going on?

  Hint: ps output in part 1.

Part 3: mdb-lookup module

Implement the same functionality as lab 7, part 2(b) in an Apache module. That is, make the Apache server respond to /mdb-lookup and

/mdb-lookup?key=search_string URLs in the same way that the http-server from lab7 did.

This time, however, for every browser request, we will make a new socket connection to the mdb-lookup-server and close the connection when we're done responding to the request. That is, we will not keep around a persistent socket connection like we did in lab 7.

Keeping a persistent socket descriptor in a static variable doesn't work (as demonstrated in part 2). Of course, there are ways to keep persistent data in the module API, but we are not going to go there in this assignment.

You may want to print things in your code for debugging. If you print things to stderr, they will show up in error_log file. You must call fflush(stderr) every time though.

The module does not take any parameter, so you will have to hard-code the host name and port number where mdb-lookup-server is running. Use "localhost" for the host name. For the port number, I wrote a little script that generates a random--but deterministic--port number based on your UNI. Log in to a CLIC machine and run the following command to obtain the port number assigned to you:

```
/home/jae/cs3157-pub/bin/get-my-port-number
```

Use that port number for mdb-lookup-server.

--

Again, your submitted patch should generate the following files (and no other) in the top-level (lab8) directory:

```
README.txt
httpd.conf
mod_helloworld.c
mod_mdb-lookup.c
```

README.txt should contain all the info that the lab asked for. httpd.conf should be a copy of the same file in apache2/conf directory that you have been modifying to alter the behavior of the server. (The TAs will build a fresh Apache server and copy your httpd.conf into the apache2/conf directory to test your configuration.) The mod_helloworld.c and mod_mdb-lookup.c are part2 and part3 code, respectively.

--

Good luck!