CS 0449 – Project 5: Multi-Threaded Web Server

Due: Tuesday, December 13, 2016, at 11:59pm

Description

Launching a browser and visiting a site on the internet involves at least two parties: the web server and the requestor (you). In this assignment, we will use pthreads and Berkley Sockets to implement a primitive web server, which talks HTTP to browsers over TCP/IP.

HTTP (HyperText Transfer Protocol) is a simple, text-based communication protocol by which a web browser requests documents from a web server, and the web server replies. For instance, if you visit a web site such as http://www.example.com/test.html, your browser does the following:

1. Connect to the IP address of http://www.example.com/test.html (obtained via DNS lookup) at port 80 (standard web server port)
2. Send the HTTP request message:

```
GET /test.html HTTP/1.0
Host: www.example.com
```

To which the server should reply (assuming it finds the file):

```
HTTP/1.1 200 OK
Date: Thu, 15 Nov 2007 00:24:55 GMT
Content-Length: 4892
Connection: close
Content-Type: text/html
```

Followed by a blank line and the contents of the file (which is 4892 bytes). If the file is not found, it should return the infamous 404 error code like so:

```
HTTP/1.1 404 Not Found
```

To get the current date, use a combination of time() and localtime() functions declared in time.h. Refer to the manpages for their usage. Here is an example piece of code:
http://www.gnu.org/software/libc/manual/html_node/Time-Functions-Example.html

Requirements

Your task is to use Berkley sockets to accept GET requests for HTML pages over HTTP. Your main thread should wait for a connection to occur, spawn off a worker thread, and have that thread communicate to the requestor according to the above protocol.

When the thread is done, it should add the request for that particular webpage to a file named stats.txt. Note that this file needs to be exclusively accessed, so you'll need to do some sort of synchronization. Each request should append something like the following to stats.txt (assuming the web client connected from IP 127.0.0.1 and port 60000):

```
GET /test.html HTTP/1.0
Host: www.example.com
Client: 127.0.0.1:60000
```

The IP and port of the client can be gathered when accepting the connection, as we've learned during the lectures.

Ports and Addresses

Port 80 is the normal web server port, but we can't all use it at the same time. Please use your designated, personal port number listed on:
http://www.cs.pitt.edu/~wahn/teaching/cs449/misc/Ports.pdf.

Please use this port and only this port. For an address of the machine, we will simply refer to it as localhost, or localhost's reserved ip address: 127.0.0.1

Testing

thoth.cs.pitt.edu is firewalled from the outside world, meaning that you will only be able to connect to your server from thoth itself. In order to do this, your best bet for testing is to use a few programs:

- telnet localhost PORT
- wget http://127.0.0.1:PORT/page.html
- links –no-connect http://127.0.0.1:PORT/page.html

telnet is a terminal emulator. You can connect directly to your server, and anything you type will be sent. This means you can manually create a GET request, and you will see the reply of the server in plain text on your screen.

wget downloads files from the internet.

links is a text-mode web browser. No graphics, no tables, and minimal font support, but it's a real, working browser.

Each of the above web clients sends HTTP requests of slightly different formats but they are all governed by the HTTP protocol. The request format is given in the below link:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html

As you can see, the request message header and message body is always separated by two CRLFs. In our simple interactions, you can assume that the web client never sends the message body. Hence, in your web server, you can safely assume that you have finished receiving an HTTP request when you receive two CRLFs. A CRLF is a sequence of a CR (ASCII code 13) and a LF (ASCII code 10) character.

My advice is to work with two SSH windows open, one with your server printing error messages to stderr, and one that you are using one of the above programs to request pages.

Submission

You need to submit:

- Your well-commented program's source

Make a tar.gz file as in the first assignment, named `USERNAME-project5.tar.gz`

Copy it to ~`wahn/submit/449/RECITATION_CLASS_NUMBER` by the deadline for credit.