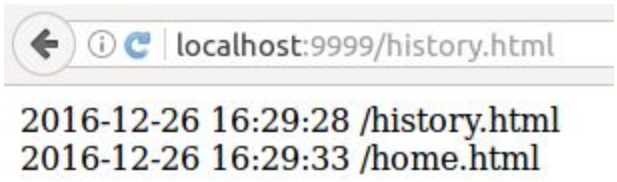
Antonina Serdyukova
CSCI 379 Computer Networking
Programming assignment #2
Due by 11:59pm, Monday, Dec. 26th, 2016



How To Run And What To Expect

My program sever.py runs a simple multithreaded web server that **serves requiested files** (which are structurally the same file really, index.html, with path repalced as per requirements, but the code is capable of serving completely different files as well).

Code is written in Python 3 and was tested on Ubuntu 16.04.1, so this is a recommended environment. To run the code:

```
python3 server.py [port number]
```

If no port number specified, server runs on port 80. Recommended ports are 4 digit ports (i.e. 9999) for avoiding `'permission denied'` errors.

To connect to the server open your browser and type:

```
localhost:[port number]
```

If no path specified, the server fetches index.html, you can also visit home page by typing:

`localhost:[port number]/home.html`

Or if you type non existing path, server will fetch 404.html file. To test:

`localhost:[port number]/blah.html`

To test **multithreading** try to connect from different browsers and incognito modes.

This server only accepts `GET` requests and fetches 404 page for all other types of request.

You will also see **information about your system and browser** on each page. It's kind of basic, but it works on all pages (index.html, home.html), except for 404 page (since it's 404, I thought this info is redundant).

You also can visit a **history page**:

`localhost:[port number]/history.html`

There you will see all the pages that were visited from your ip with a corresponding time stamp. It's far from ideal, I store the history logs in a file named as a corresponding IP address. For instance, you will see ip-log.txt file that will have all the IPs ever connected to the server, in my case there is the localhost IP in my ip-log.txt. Then a file 127.0.0.1 is created and that is where all the activity associated with this IP is recorded. Then history information for the current IP is displayed on /history.html.

There are definitely better ways to do this, but with limitations on the use of libraries, I could just built this.

Technical Details*

Connection is established via TCP/IP socket using the socket library by creating a socket, binding and listening on the specified port.

Increased request queue size is achieved by passing large `BACKLOG` argument to the socket's `listen()` method.

Multithreading is achieved via system's `fork()` method that produces two returns one in the parent process (returns child's PID) and one in the child process (0). The `fork()`

method clones parent's file descriptors to the child, therefore even when the parent's connection is closed, the child's socket is kept alive. Magic!

The child copy closes the parent's listen socket because the child won't be accepting any new connections, but will only process requests from the same connection. Only children handle client requests, parent's role is to create children.

To prevent children to hang in the memory failing to close their connection within the parent, we use `signal handler` and `waitpid` system (with use of `WNOHANG` option to prevent signal flood). On exit in a child process, the kernel sends `SIGCHLD` signal, then the parent catches this event and waits for the termination signal.

To avoid the parent being locked in `accept()` call interrupted with `EINTR` error, we restart the `accept()` system call.

The system info is extracted from the request header and matched upon very basic mapping data to determine the client's browser and OS.

History functionality is based on IP address recognition. See more details in the 'How To Run And What To Expect' section.

* Special thanks to Ruslan on ruslanspivak.com and to you Professor for pointing out such a great resource.