Project 2

Multi-Threaded Web Proxy

# GitHub Repository

https://github.com/skmathew/Multi-Threaded-Web-Proxy---Group1

# Problem

For Project 2 we needed to create a multi-threaded web proxy according to the following requirements.

Clients needed to be able to call the proxy server from a web browser in the following format:

* Template: <IP address of proxy server> : <port number of proxy server> / <site being accessed>
* Example: 129.120.151.94:9003/cse.unt.edu

The proxy server needed to receive client HTTP requests, generate a new HTTP request and send it to the host, receive the requested page from the host, and finally send the page back to the client. The web proxy also needed to have the following features:

* Block websites found on a blacklist
* Filter inappropriate language from the webpage
* Cache sites that have already been visited

# Approach

We collaborated on the project on GitHub. We used the provided code in *serverThreads.c* and *ProxyServer.py* as a template. We designed the proxy server to handle storing pages in the cache by reading and writing the page content files. This also made it easier to filter inappropriate language from the file by reading through the file and substituting any matching tokens. The blacklist and list of inappropriate words are also stored in text files.

# Solution

The proxy server begins by setting up a socket on port number defined as LISTEN\_PORT to listen for client connections. For each accepted client connection, the server creates a new thread and gives control to the client\_handler.

The client\_handler does the following:

1. Calls blocked\_websites to check if the requested page is on the blacklist
2. Checks if the requested page is already in the cache
   * If the page is not already in the cache, calls GET\_page (forms the HTTP request, sends the request to the host, receives the requested page, and saves the page to its own file)
3. Opens the saved page in the cache
4. Calls str\_replace to filter out inappropriate language from the page
5. Sends the page to the client

The user remains connected while the socket processes the user message. Then the thread is terminated.

# Screenshots

## Compile Proxy Server

ProxyServer.c successfully compiles on cse01.cse.unt.edu (129.120.151.94) on port 9003. After executing, the proxy server waits for client connections.

## Call Proxy Server from Web Browser

The client calls the proxy server from the Chrome web browser, requesting the page “”.

129.120.151.94:9003/””

## Proxy Server Receives Request

The proxy server receives and processes the client request.

## Client Receives the Page

The client correctly receives the page.

## Proxy Filtered out Inappropriate Language

The proxy replaced the word “” with “”.