Assignment Two
Webserver

Set: *5th of May 2014*
Due: *15th of May 2014 @ 23:55 CEST*

Synopsis:
Implement a HTTP compatible webserver that can serve static files to a browser.

Introduction

This is the second of four assignments in the *Datanet* course. The assignments are pratical in nature, and will give you you a hands-on approach to some of the technology that we all use and depend on as a part of our daily life.

In this second assignment you must implement a HTTP compatible webserver, using socket programming. The webserver will only support a subset of the HTTP protocol, but will be able to serve files.

Before you start on this assignment, you should read and understand the rest of the assignments as they are based on this one.

Implementation

You must implement a web server that is capable of serving files from a local folder. This means that you are only required to implement the HTTP GET method, but you may want to implement the HEAD method as well, as it may be useful in the later assignments. You should make sure that your implementation is flexible enough to allow for later extension with additional commands.

The server should work as a plain file server and should accept a path to a folder and a port number. The server may also take an address to listen on, but if this is not implemented, the server should listen on all addresses on the port specified. Files and folders in specified path should be served up by the webserver. If a folder on the server is accessed (with a URL ending in a `/`) then you should serve up the contents of file named `index.html`, if one exists, or a listing of the files and folders, if `index.html` does not exist.

When a request is received, it should be parsed and verified as a valid HTTP request. The URL from the request is then mapped to the underlying file system. If the requested file does not exist, the server responds with error code 404 and a short message, otherwise the file contents are returned. You will need to use other error codes as appropriate.

In general you should familiarize yourself with HTTP RFC (RFC 2616), which explains how the protocol is supposed to be interpreted. Be aware that the HTTP protocol is very large, and you should by no means attempt to support all features. If something is overly complicated to implement, you should most likely not be implementing it. If you are in doubt, please use the Absalon forum to ask for clarifications, but try to read the RFC before asking questions.

The features you implement should be implemented such that they are standards compliant. You should also ensure that your server is standards compliant even given the features you do not implement. As an example, you do probably not want to implement persistent connections, but **you must** send the appropriate header to your clients indicating that you do not do so.

You can implement the web server in any reasonably readable language (e.g. Python, Java, C#, SML, Perl, ...). Your implementation *MUST* use socket programming, meaning that the use of an HTTP aware library is not allowed. You can use other libraries, e.g., text parser, regular expressions, etc. Your web server should be able to process more than one request at a time. You should use an appropriate library or feature of the language to implement the serving of multiple requests.

For Python you might use an asynchronous library such as *Twisted*, *asyncor* or even the *select* library. You can also use threads, using the *ThreadingMixIn* of the *SocketServer* framework. Similar facilities exist for other languages.

If you are in doubt, please use the Absalon forum to ask for clarifications.

Your Report

Your report *MUST* be written in ACM format. An ACM template for LaTeXand Microsoft Word is available for download via Absalon.

Your reports should contain:

- An abstract describing the contents of your report
- A description of your web server design (including libraries and frameworks used)
- A description of your web servers limitations
- A description of what tests you have performed and their outcomes
- A description of supported headers, and an explanation of why you support those
- A description of how your implementation is prepared to deal with the following assignments

Deliverables for This Assignment

You are encouraged to work in informal groups for this assignment, for the purpose of discussing implementation details and limitations. We strongly encouraged you to come to the exercise classes, where we will use time to discuss the design, implementation etc. The implementation and report that you hand in must however be **your own individual work**.

You should submit the following items:

- A single PDF file, A4 size, no more than 3 pages, in ACM format, describing each item from report section above
- A single ZIP/tbz2 file with all code relevant to the implementation

Handing In Your Assignment

You will be handing this assignment in using Absalon. Try not to hand in your files at the very last-minute, in case the rest of the Datanet students stage a DDoS attack on Absalon at the exact moment you are trying to submit. **Do not email us your assignments**.

Assessment

The assignment will be scored on a scale of 0-10 points. There will be **no resubmission** for this assignment. In order to participate in the exam, you must obtain a total of 24 points over the four assignments.