School of Computing and Information Systems
COMP30023: Computer Systems
Assignment 1
**Due date: No later than 11:59pm on Thursday 19th April 2018**
Weight: 15%

## Project Overview

The aim of this project is to increase your familiarity with socket programming, multi-threading, and the HTTP protocol. Your task is to write a basic HTTP server that responds correctly to a limited set of GET requests. The HTTP server must return valid response headers for a range of files and paths.

Your HTTP Server must be written in C. Submissions that do not compile and run on a NeCTAR instance may receive zero marks. You must write your own HTTP handling code, you may not use existing HTTP libraries.

## Project Details

Your task is to write a simple HTTP Server. There is only a limited range of content that needs to be served, and it only needs to respond to GET requests. The content that needs to be served is as follows:

| Content | File Extension |
|---|---|
| HTML | .html |
| JPEG | .jpg |
| CSS | .css |
| JavaScript | .js |

You can assume that all files have a valid file extension.

The minimum requirement is that the server implements HTTP 1.0, and as such, your server does not have to support pipelining or persistent connections. Your server should be able to handle multiple incoming requests by making use of Pthreads (or similar concurrent programming technique) to process and respond to each incoming request.

Your server program must uses the following command line arguments:

- port number
- string path to root web directory

The port number will constitute the port the server will listen for connections on. The string path to the root web directory points to the directory that contains the content that is to be served. For example:

"/home/comp30023/website"

All paths requested by a client should be treated as relative to the path specified on the command line. For example, if a request was received with the path */css/style.css* it would be expected to be found at:

"/home/comp30023/website/css/style.css"

The server must support responding with either a 200 response containing the requested file, or a 404 response if the requested file is not found. You can assume that request headers will be valid, but paths to non-existent files may be requested, and if they are a 404 should be returned. Response headers should be valid and must included at a minimum:

- Http Status
- Content-type

### Program execution / command line arguments

To run your server program on your NeCTAR cloud instance prompt:

```
./server [port number] [path to web root]
```

where:

- `[port number]` is a valid port number (e.g., 8080), and
- `[path to web root]` is a valid path (e.g., /home/comp30023/website)

## Submission details

Please include your name and login id in a comment at the top of each file. Our plan is to directly harvest your submissions on the due date from your provided GitLab git account. You must create a repository called `comp30023-2018-project-1` in your GitLab account and push your submission to it.

You must submit program file(s), including a Makefile. Make sure that your Makefile, header files and source files are added/committed and pushed to your GitLab repository. Do not add/commit object files or executables. Anything you want to mention about your submission, write a text file called README.

- If you do not use your Git repository for the project you will NOT have a submission and will be awarded zero marks.
- It should be possible to "checkout" the Git repository, then type `make clean` and then `make` to produce the executable `server`.
- Late submissions will incur a deduction of 2 marks per day (or part thereof).
- If you submit late, you MUST email the lecturer, Chris Culnane cculnane@unimelb.edu.au.

**Extension policy:** If you believe you have a valid reason to require an extension you must contact the lecturer, Chris Culnane cculnane@unimelb.edu.au at the earliest opportunity, which in most instances should be well before the submission deadline.

Requests for extensions are not automatic and are considered on a case by case basis. You will be required to supply supporting evidence such as a medical certificate. In addition, your git log file should illustrate the progress made on the project up to the date of your request.

**Plagiarism policy:** You are reminded that all submitted project work in this subject is to be your own individual work. Automated similarity checking software will be used to compare submissions against each other and known public source code. It is University policy that cheating by students in any form is not permitted, and that work submitted for assessment purposes must be the independent work of the student concerned.

Using Git is an important step in the verification of authorship. You are encouraged to commit regularly so that you have a record of your work. This is also best practice when using version control software.

## Assessment

Code that does not compile and run on a NeCTAR will be awarded zero marks. Your submission will be tested and marked with the following criteria:

- Compiles from Git (3 marks)
  - Code successfully added to your git repository
  - Make file included (and it works)
  - Clarity of code – appropriate comments and documentation where necessary
  - Code correctly implements command line interface
- Server runs and sends valid responses (4 marks)
  - Code runs on a NeCTAR instance without seg faulting
  - Server sends a valid HTTP 200 response in reply to a GET request for an HTML file located in web root directory
  - Server sends a valid HTTP 404 response in reply to a GET request for a non-existent file in the web root directory
- Server MIME types and paths (4 marks)
  - Server sends a valid HTTP 200 response with the correct MIME type in reply to a GET request for either a JavaScript or CSS file located in web root directory
  - Server sends a valid HTTP 200 response with the correct MIME type in reply to a GET request for a JPEG file located in web root directory
  - Server sends a valid HTTP 200 response with the correct MIME type in reply to a GET request with a path below the web root for any of the specified file types (e.g. `GET /css/style.css HTTP/1.0`)
- Multi-threading (4 marks)

  - Server uses Pthreads (or similar concurrent programming technique) to process incoming requests and sending responses
  - Server can process and respond to at least two HTTP requests concurrently

**Testing your server**

Your server will be partially evaluated using automated testing. A sample test script is provided in the repository:

`https://gitlab.eng.unimelb.edu.au/cculnane/COMP30023Assignment1TestScript`

The repository contains a shell script and sample HTML, CSS, JavaScript, and JPEG files. You may use this script to test your server successfully passes the following assessment criteria:

- Server runs and sends valid responses (4 marks)
- Server MIME types and paths (4 marks)