CMPUT 379 Assignment 2, Client Request Format

The simplified HTTP 1.1 protocol for this assignment is relatively straightforward. The server must listen to the desired port, and service an HTTP 1.1 connection each time a client connects to it. The client is expected to send a request, and the server must send either a successful response, or an error response. For your assignment, your server is only required to implement the GET request and response, with a few key error messages in certain situations. HTTP requests and replies are quite simple, they consist of multiple lines, where each line is terminated by a line feed (\n) OR a carriage return and linefeed together (\r\n). A request, or reply, consists of:

- An initial line (The request, or the response code)
- Zero or more header lines (Header: Header Value)
- A blank line (\n, or \r\n on a line by itself)
- Optional message data (in your case, the contents of a file, or an error message)

A **request** will come from the client as multiple lines of input, and look like this:

```
GET /someplace/file.html HTTP/1.1
From: someuser@somewhere.org
User-Agent: Bloatzilla/7.0
[blank line here]
```

This is a request from the client to obtain the file "`someplace/file.html`" inside the document directory specified on startup. The only part of the request your program need worry about is the GET line, and then discard everything up to, and including, the blank like at the end of the request. (You must however, notice that the request is correctly terminated by a blank line). Notice also that the leading slash "/" character in the GET request refers to the directory of where the documents are stored (it is *not* the root directory of the file system). In this sense, the file names in the GET request are relative to the server's documents directory. So, if for example your server's documents directory is `/some/where/documents` then the request in the above example is for the document `/some/where/documents/someplace/file.html`

Note that on GET requests, for the purposes of this assignment your server may ignore all the header data on the requests. You will need to correctly test that the request is terminated by a blank line.

Notice that because of its compatibility with the HTTP/1.1 protocol, your servers should be able to cope fine with requests sent by your browser of choice. That is, you should be able to

test your servers by using any respectable web browser. For example, assuming you are using the browser on the same host that your server is running, the request can be sent by the following URL: `http://localhost:PORT/someplace/file.html` where PORT is the port on which your server is listening.

Assuming the server can send the file "`someplace/file.html`" the following **response** will be sent to the client:

```
HTTP/1.1 200 OK
Date: Mon 21 Jan 2008 18:06:16 GMT
Content-Type: text/html
Content-Length: 1234

This is the contents of file.html
it would continue for exactly 1234 bytes
```

Your server can always return "`Content-type: text/html`" - it must return the correct Date, and Content-Length header on all replies. There is a blank line between the end of the headers and the start of the file. Your server may assume that anything after the GET and the leading / in the request is the name of a file to be retrieved - you do NOT need to support any arguments for CGI scripting or any other strange things you may have knowledge about HTTP. All your servers need do is to serve files with GET, and you may safely assume that all valid requests will start with GET, have a path to a file starting with a leading /, and then end with HTTP/1.1. A GET with multiple things between the GET and the HTTP/1.1 can be considered a Bad Request as described below.

If the server receives a request it does not understand (Anything not a GET request, or not ending in " HTTP/1.1") It must return a BAD REQUEST error, which can look exactly like this:

```
HTTP/1.1 400 Bad Request
Date: Mon 21 Jan 2008 18:06:16 GMT
Content-Type: text/html
Content-Length: 107

<html><body>
<h2>Malformed Request</h2>
Your browser sent a request I could not understand.
</body></html>
```

If the server receives a request for a file that does exist, it must return a Not Found error - which can look exactly like this:

```
HTTP/1.1 404 Not Found
Date: Mon 21 Jan 2008 18:06:16 GMT
Content-Type: text/html
```

```
Content-Length: 117

<html><body>
<h2>Document not found</h2>
You asked for a document that doesn't exist. That is so sad.
</body></html>
```

If the server receives a request for a file that is not permitted to be read, it must return a Forbidden error - which can look exactly like this:

```
HTTP/1.1 403 Forbidden
Date: Mon 21 Jan 2008 18:06:16 GMT
Content-Type: text/html
Content-Length: 130

<html><body>
<h2>Permission Denied</h2>
You asked for a document you are not permitted to see. It sucks to be you.
</body></html>
```

If the server has some sort of unplanned error while servicing a request, it must attempt to return a Server Error message - which can look exactly like this:

```
HTTP/1.1 500 Internal Server Error
Date: Mon 21 Jan 2008 18:06:16 GMT
Content-Type: text/html
Content-Length: 131

<html><body>
<h2>Oops. That Didn't work</h2>
I had some sort of problem dealing with your request. Sorry, I'm lame.
</body></html>
```

The full implementation of the HTTP 1.1 protocol is possible, but is not required for this assignment. For more information about HTTP 1.1, please see http://www.jmarshall.com/easy/http/. Please remember that you are *NOT* expected to implement the entire HTTP 1.1 protocol in your servers, only expected to implement GET for files and error handling as outlined above.