CS118 - Spring 2015
Project 1: Concurrent Web Server using BSD Sockets

Nicolas Langley 904433991
Partner: Michael Levin 302818371 Login: levinm

May 3, 2015

1 High Level Server Model Overview

The goal of this project is to implement a web server. In part A requests are output to the server console, in part B requests are retrieved and sent back to the client and displayed in the web browser, which requires parsing the incoming request as well as coding a mechanism to create the response including the header and data.

The server begins by creating a socket using the types imported from the sys/socket.h file. We set the port number and then bind the socket to the server address. The socket then begins listening for requests. When a request is received the listener forks a new process for it and continues listening.

When the process forks it passes the socket information and the request. The request is parsed for the file path and and content type. The header data for the file is generated and if the requested type is html then the data is text and it is encoded in ASCII. If the requested type is GIF or JPEG then binary data is not reformatted but simply loaded into a character array for transport.

The date and time are recorded and packaged together with the file information into the header. The header is passed to the socket and then the file data is passed to the socket.

2 Difficulties Encountered and Solutions

One of the first problems we needed to solve was the formatting of the headers. We needed to understand the format of the request header in order to parse the information and access the appropriate file. We also needed to understand the format of the sender header in order to properly send the requested file. In order to process the request header we had to parse the string and identify the substring containing only the path to the file being requested.

The sender header required lots of information about the file being requested. We had to determine the last modified time as well as the size of the file and also obtain the current time to input into the Date Sent field for the header. This required us to include additional functions and types using files like sys.stat and stat.h.

When sending data we had to determine the type of content that was being requested. Depending on the content type, the file must be loaded differently. The distinction is made between text and binary files. We had a challenge loading the binary files due to an accidental mistake in computing the length of the header. This resulted in a undesired byte being transferred as part of the binary file causing it to be unreadeable. This problem was a result of the write buffer to the socket not being cleared between sending the header and the data.

3 How to Compile and Run Code

The code is based off of the samples provided. This means that in order to compile and run the code the following steps must be followed:

1. Run the Makefile within the /src directory NOTE: -libstdc++ has been added has a linker flag (may be OSX specific)
2. Run the resulting serverFork executable with port number as argument

4 Sample Outputs

For testing the server, we used 3 test files (one for each content type). These files are shown below:

The first file is a simple HTML file:

```
<!DOCTYPE html>
<html>
  <head>
      <title>CS 118 Test Page</title>
  </head>
  <body>
    Test page for CS 118 Project 1
  </body>
</html>
```

The second file is a JPEG file:

Lastly, we have a GIF file: