Assignment 4

Assigned : November 19, 1998

Due : December 11, 1998

Simple Web Server **[30 Marks]**

You are to write a simple, multithreaded web-server. The main thread of the server sets up an INET socket and listens for connections from web-browsers. As connections are made, a PTHREAD thread is spawned to handle the request. The thread terminates once each request is completed. We will cover PTHREADS in class starting next week.

Synopsis

```
myWebServer [rootPath] [-p portNum]
```

You invoke the server with the command "myWebServer". There are two optional arguments. The parameter rootPath is an absolute or relative path that is searched for the desired files to be served. If `rootPath` is not specified, it defaults to the current directory. The `portNum` parameter is used to specify the INET socket port that the web-server listens on. The default value of this port is 3000 plus your UID on CDF.

Basic Operation

Your web-server will receive HTTP requests from web browsers (it is recommended you use Netscape for testing). The first line of an http request is

```
GET <pathName> HTTP/1.0
```

A web browser will also send additional lines in the request, but these can be ignored for the purpose of this assignment. The `pathName` will look like an absolute path, but should be treated as being relative to the `rootPath`. For example, if `rootPath` is "/u/maclean/a4root", then a `pathName` of "/Docs/info.html" is mapped to the file "/u/maclean/a4root/Docs/info.html".

Each time the server's main thread accepts a connection, it spawns a child thread to handle the request. The child attempts to find the file, and if found it sends the appropriate HTTP header and then the file's data.

The following sample HTTP request headers may be of use:

In response to "GET ~/maclean HTTP/1.0"

```
HTTP/1.1 301 Moved Permanently
Date: Thu, 19 Nov 1998 13:14:10 GMT
Server: Apache/1.2.0
Location: http://www.cs.toronto.edu/~maclean/
Connection: close
Content-Type: text/html

<HTML><HEAD>
```

```
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.cs.toronto.edu/~maclean/">here</A>.<P>
</BODY></HTML>
```

In response to "GET /~maclean/ HTTP/1.0"

```
HTTP/1.1 200 OK
Date: Thu, 19 Nov 1998 13:15:38 GMT
Server: Apache/1.2.0
Last-Modified: Wed, 22 Jan 1997 19:01:30 GMT
ETag: "7ad66-2ac-32e6640a"
Content-Length: 684
Accept-Ranges: bytes
Connection: close
Content-Type: text/html
```

HTML data here

In response to "`GET /~maclean/redKnot.gif HTTP/1.0`"

```
HTTP/1.1 200 OK
Date: Thu, 19 Nov 1998 13:17:53 GMT
Server: Apache/1.2.0
Last-Modified: Wed, 28 Aug 1996 20:08:27 GMT
ETag: "7ad8a-14c-3224a73b"
Content-Length: 332
Accept-Ranges: bytes
Connection: close
Content-Type: image/gif
```

image data here (binary format, use `write()` *not* `fprintf()`*)*

In response to "`GET /~maclean/knot.jpg HTTP/1.0`"

```
HTTP/1.1 200 OK
Date: Thu, 19 Nov 1998 13:19:25 GMT
Server: Apache/1.2.0
Last-Modified: Fri, 15 Mar 1996 19:36:49 GMT
ETag: "7ad69-383-3149c6d1"
Content-Length: 899
Accept-Ranges: bytes
Connection: close
Content-Type: image/jpeg
```

image data goes here (binary format, use `write()` *not* `fprintf()`*)*

In response to "`GET /~maclean/foo.html HTTP/1.0`"

```
HTTP/1.1 404 File Not Found
Date: Thu, 19 Nov 1998 16:42:03 GMT
Server: Apache/1.2.0
Connection: close
Content-Type: text/html

<HTML><HEAD>
<TITLE>404 File Not Found</TITLE>
</HEAD><BODY>
```

```
<H1>File Not Found</H1>
The requested URL "/~maclean/foo.html " was not found on this server.<P>
</BODY></HTML>
```

A sample HTTP header returned by your server is as follows:

```
HTTP/1.1 <status code> <status message>:
Date: <date string>
Server: myServer/0.99
Location: <new location, only if status code = 301>
Last-Modified: <last-modified date string>
Content-Length: <number of bytes in file being served>
Accept-Ranges: bytes
Connection: close
Content-Type: <content type>
```

served file data

Legal status codes are 200 (implies status message = "OK"), 301 (implies status message = "Moved Permanently"), and 404 (implies status message = "File Not Found"). The date string should be in the format shown in the samples above, and should be in **Greenwhich Mean Time** (GMT). Ditto for the last-modified date string. Legal values for content type are "text/html", "image/gif" and "image/jpeg", depending on the type of file to be served. You may assume that you will only be asked to serve files with the extensions `.html`, `.gif` and `.jpg` so that the file types may be easily determined. When the status code is 301, you will return the location of the re-direct and omit the "Accept-Ranges", "Content-length" and "Last-modified" tags.

All HTTP headers terminate with two `'\r'` characters in succession. New lines are marked with `'\r'` instead of `'\n'`. When writing data back to the client, you should use `write()` so that it will be easy to send binary data. In fact, you should treat all files (even `.html` files) as if they are binary data.

Note that directory requests are expected to have a trailing `'/'`. If you receive one that does not, you should add the trailing slash and return a redirect message as shown above. If a directory is specified instead of a file, your web server will return `index.html` in that directory if it exists, or `404 File Not Found` if it does not.

For more detailed information on the HTTP protocol, refer to the following:

http://www.sunsite.auc.dk/RFC/rfc/rfc1945.html

http://www.sunsite.auc.dk/RFC/rfc/rfc2068.html

Statistics Collection

The main thread will maintain count of 1) the total number of connections processed, 2) total number of files served, 3) total bytes of files served. Each thread will update the master statistics record: in order to avoid corrupting this record the server will establish a semaphore to mediate access to this data structure. The main process will write the statistics to the screen every 5 minutes.

Bonus [5 Marks]

Extend your simple web-server to process CGI programs as well. You may assume the following: 1) CGI programs will always have names that end in .cgi, 2) any CGI program is executable "as is", *i.e.* it is a compiled program or a shell/Perl script whose 1st line is `#!<path of interpreter to execute>`, 3) any CGI program will write the required HTTP header information as part of it's output, and 4) there will be no parameters passed to the CGI program by the browser. You must detect the case where there is a failure `exec()`-ing the CGI program, and write the appropriate response to the user (you will need to research just what the appropriate response is). If you receive a request for a CGI program which does not exist, return `404 File Not Found`.

What to Submit

1. All `.h` and `.c` files associated with your web-server. You are encouraged to use separate header and `.c` files as required to make your project more manageable.

2. You are required to submit a makefile that properly creates an executable version of your program on CDF. It should support a "clean" option that removes the .o and executable files.

3. You must submit these files both electronically and via hard copy. Failure to submit hard copy will result in a 50% penalty. There must be a comment at the top of each file with your name and student number.

4. You must submit (electronically only) an executable version of your code.

5. For those attempting the bonus section, you must clearly indicate this at the top of your hard copy! Failure to do so will result in the bonus section not being marked.