Project 2: Retrieving WWW documents

1. Hypertext Transfer Protocol (HTTP)
   Protocol between browsers and Web servers to transfer resources.
   — for more detail see HTTP Made Really Easy at http://www.jmarshall.com/easy/http/

   (a) Uniform Resource Location (URL)
   `http://www.somehost.com:80/path/file.html`

scheme	http
host	www.oreilly.com
port	80
resource path	/path/file.html

   (b) Structure of HTTP Transactions
   - The Request and Response Cycle
     - Web Browser
       The browser opens a TCP connection at specified port (default 80), sends the request to the server over that connection.
     - Web Server
       The server processes the request and returns a response message
   - The Format of the Request/Response Messages
     - Header/Body

```
<request method line or response status line>
Header1: value1
Header2: value2
Header3: value3

<optional message body goes here, like file contents or
query data; it can be many lines long, or even binary data>
```

     - Browser Requests
       * Common HTTP Request Methods
         GET, *POST*, *PUT*, *DELETE*
       * Common HTTP Headers

Content-Lenght	the length (in bytes) of the request content
Content-Type	media type of the request
Accept	browser name, version, etc.
User-Agent	browser name, version, etc.

       * An Example

```
GET /index.html HTTP/1.0
User-Agent: Mozilla/1.1N
Accept: text/*, image/gif, image/jpeg, */*
```

     - Web Server Responses

  - Status Line
    `HTTP/1.0 200 OK` *or*
    `HTTP/1.0 404 Not Found`
  - Common Status Codes
    - 200 OK
    - 404 Not Found
    - 301 Moved Permanently
    - 302 Moved Temporarily
    - 500 Server Error

(c) A Manual Transaction with `telnet`

```
% telnet www.bookpool.com 80
Trying 209.113.145.3...
Connected to refugio.bookpool.com.
Escape character is '^]'.
GET /index.html HTTP/1.0
User-Agent: Mozilla/1.1N
Accept: text/*, image/gif, image/jpeg, */*


HTTP/1.1 200 OK
Date: Thu, 19 Oct 2000 22:26:13 GMT
Server: Apache/1.3.14
Last-Modified: Sat, 05 Feb 2000 16:08:06 GMT
ETag: "0-11d-389c4ae6"
Accept-Ranges: bytes
Content-Length: 285
Connection: close
Content-Type: text/html

<HTML>
<HEAD>
  <TITLE>Netscape Enterprise Server 4.0</TITLE>
</HEAD>
<FRAMESET BORDER=0 ROWS="80, *">
<FRAME NAME="banner" SRC="banner.html" SCROLLING="no">
<FRAME NAME="content" SRC="launch.html" SCROLLING="auto">
</FRAMESET><noframes></noframes>
</HTML>
```

2. Project Requirement

   (a) Write a HTTP client program which will use the `GET` request to download a document from any Web Server.

   (b) The command line format of the program is
   `% getpage domain-name/path`

The program will send a GET request to the server, and read back the server response. If the status code of the response is “200 OK”, the program will print the body of the response on the standard output; otherwise the program should print an error message together with the response status code.

3. A Sample Implementation (*may not work*)

   (a) Open a TCP connection to the server at port 80

   (b) Construct a GET request and send it to the server

```
char    request[1024];
char    buf[1024];
FILE    *fp;
char    *len_label = "Content-Length:";

/* you may check if the request is too long */
snprintf (request, sizeof request, "GET %s HTTP/1.0\r\n\r\n", filepath);
write (sock, request, strlen (request));

/* tell the server the client will never send more data */
shutdown (sock, 1);

/* turn the socket into a C standard file stream for reading */
fp = fdopen (sock, "r");

/* get the status line */
data = fgets (buf, sizeof buf, fp);
if (data == NULL) {
    /* error process */
}
/* handle the status code in the status line */
...

length = ULONG_MAX;
/* process the header part of the response */
while (1) {
    data = fgets (buf, sizeof buf, fp);
    if (strncasecmp (buf, len_label, strlen (len_label)) == 0) {
        /* get the length of the document */
        length = strtol (...);   /* or use atoi for simplicity */
    }

    /* a blank line ends the header part */
    if (buf contains a blank line)
        break;
```

```
}

total = 0
while (total < length) {
    n = fread (buf, 1, sizeof buf, fp);
    if (n == 0) {
        if (ferror (fp)) {
            /* some error occurred in reading */
            ...
        }
        break; /* otherwise we reached the EOF */
    }
    write (0, buf, n); /* print the data on standard output */
    total += n;
}
fclose (fp);  /* this will also close the socket */
```

4. Grading
   - the program runs correctly 75%
   - overall design quality 15%
   - coding format and comments 10%
5. Some requirements
   (a) Each student should work on this project **individually**.
   (b) **Both hard copy and electronic copy** should be handed in.
   (c) The file name of the executable should be `getpage`. The program gets the URL from the command line and prints the result to the standard output.

   ```
   % getpage http://www.cs.twsu.edu/index.html > result
   ```

   You can compare your result with that generated with the command `lwp-download`. Your program should download both text files and binary files.
   (d) **Due date is Dec. 3.**