Project 2: HTTP download using "Range"

Type of Project: Individual or Group of Two
Deadline: 2017-03-01, 11:59pm
Language: C/C++
Points: 20 points
Submission: via ELC

Submission Guidelines: Submit through ELC, as usual. Submit ONLY the source code (zip or tar.gz file). Include a "**Makefile**", so that the code can be compiled simply by executing 'make'. Also, make sure to name the output of the compiler, i.e., the program file name, as "http_downloader." All files need to be under the same main directory (i.e., do not create subdirectories under "**LastNameFirstName**-http_downloader").

NOTE: project submissions that do not follow the guidelines risk to receive 0 points
.

Project Description: In this project, you are required to write a program that takes in input (on the command line) the URL of an object to be downloaded, and the number of connections through which different parts of the object will be retrieved using "Range:". The downloaded parts need to be re-stitched to compose the original file.

For example (notice that the line below is an actual example of how your program must be launched)

$ ./http_downloader http://cobweb.cs.uga.edu/~perdisci/CSCI6760-S11/Project2-TestFiles/topnav-sport2_r1_c1.gif 5

will open 5 TCP connections to cobweb.cs.uga.edu on port 80, and retrieve the .gif file in 5 parts of approximately equal length. Finally, the program will put the parts together and write the output into a file called "topnav-sport2_r1_c1.gif". You should name the files containing the parts of downloaded content as "part_i", where i is an index. In the example above, the program will output the parts into 5 different files called "part_1", "part_2", ..., "part_5", along with the reconstructed "topnav-sport2_r1_c1.gif" file. DO NOT delete the "part_i" fiels after you are done recomposing the original file.
Save all downloaded files into the same director "./" **from which the program is launched** (do not create any subdirs).

To make sure your software downloads and correctly reassambles objects from the web, you can use *md5sum* to compare your result with the original file downloaded using a browser (or *wget* or *curl*), for example.

NOTE: You don't need to worry about handling Server "errors" (e.g., redirections, unavailable Range option, etc.). I will only test your software on objects retrieved from websites that support the Range option, and for which no special error handling is required. Of course, make sure to use HTTP/1.1 and that your HTTP requests are correctly formatted, otherwise they will fail even the simpler tests.

HINT: To divide the object to be retrieved into approximately equal parts, you first need to retrieve the length of the object without retrieveing the object itself. One way to do this is using a HEAD request and parsing the Content-Length field in the response. An alternative (optional) way to do this is by using a GET request with a particular value in the Range field (I will leave it to you to figure this out, if you decide to use this second

option).

TESTING YOUR CODE

You can use the following URLs to test your code:

http://cobweb.cs.uga.edu/~perdisci/CSCI6760-S11/Project2-TestFiles/topnav-sport2_r1_c1.gif
http://cobweb.cs.uga.edu/~perdisci/CSCI6760-S11/Project2-TestFiles/Uga-VII.jpg
http://cobweb.cs.uga.edu/~perdisci/CSCI6760-S11/Project2-TestFiles/story_hairydawg_UgaVII.jpg

Make sure that when you recompose the output from the different parts, the md5sums matches the following ones:

04e1f00315854f382d00311c599a05f8 story_hairydawg_UgaVII.jpg
0592796fa4f06f806e5d979d7e016774 topnav-sport2_r1_c1.gif
9dc5407cc368aaaa33c6cc2c284ab5c4 Uga-VII.jpg

I suggest you to also test your code on other URLs chosen by yourself, and try to determine if there are any websites that support the Range option but cause problems to your code.
You should open multiple connections in parallel, and let different Threads handle each of these parallel connections.
To split the download in multiple threads you can use the following functions:

#include <**pthread.h**>

int **pthread_create**(pthread_t *restrict thread, const pthread_attr_t *restrict attr, void *(*start_routine)(void*), void *restrict arg);
int **pthread_join**(pthread_t thread, void **value_ptr);

On your VM, you might need to install make and gcc, if you have not yet done so, using apt-get install.