CS3505 PS3 (Web Crawlerman)

By Team Amurica

Carlos Brenneisen Ben Nelson Tucker McKnight Scott Stramberg

2/8/11

**Section One:**

The web crawlerman project will visit websites and parse their html information into an object readable by the program. It will then phase out the information that is not necessary (such as css, php, javascript and any other scripting or non html based information we find on the page). Afterwards it will search through the html to find every ‘a href =…’ tag and then save the links in another object which will be used to visit after the search of this page is completed. The page will then continue to be parsed and words will be stored in a data structure along with their count. Then after all this is completed the web crawlerman will visit each site stored in the links object recursively performing the same operations.

A count will be maintained for each website visited and parsed as well as the count of each word that is used. An object will be used to make sure no website is repeatedly visited. Afterwards the counts that are maintained will be merged together and the finished count will be outputted as a file. This file will be a simple text file with statistics on how long it took to run the algorithm, how long it took to visit each web page and the word count statistics including the top fifty words that were used.

**Section Two:**

* lib curl : We will use the networking libraries provided by the curl library to perform all networking operations as needed.
* Hash Table: We will use the hash table to maintain key-value pairs of words used in the document and their respective counts.
* Graph: We will use a graph structure that will be built to maintain a map of the websites visited and to make sure no websites are visited more than one time.
* Graph Traversals: We will use these to visit the websites within our graph class.
* Input: We will parse the file into a html\_doc class with this class, this will phase out any other scripts present other than html script and words in the document
* Html\_doc: This will be the basis of iterating through the document and finding out what words are present.
* Output: We will output the file into a text based document with statistics listed

**Section Three:**

We will implement an algorithm to collate all the web pages images into a single collage. This algorithm is doable because for each web page you would just maintain the link to the images source and maintain that in a separate object which would then be called when everything needed be collated. You wouldn’t actually be downloading every image; you would just be processing the images together so it should be doable. Even if it isn’t for every web page we could still do it for the top xyz amount of images too and that would still be pretty cool. This is interesting because pictures are cool.

We will also implement an algorithm that will return the website address for a website containing a certain amount of a certain word. This will be something akin to a search engine. This is doable because it is pretty basic searching and sorting of a document and is interesting because it makes us feel like we’re making google.

**Section Four:**

Major Components:

Fixing Old Code 2/11/11

Input 2/12/11

Output 2/13/11

Parser 2/15/11

Graph 2/17/11

Searching/Traversals 2/19/11

Threading magic 2/20/11

(Dates to be changed as needed for members of the team to be accommodated)

Meeting One: Revised Plan and UML Documents

Meeting Two: Use Cases and Test Planning

Meetings Three Onward: Code

Meeting Last – 1 : Review of Project and Team Member Performance

Meeting Last: Final Report and Reflections

We will implement the project through the use of Extreme Programming using pair programming. One person in the pair will work solely on coding tests for the planned assignment while the other person in the group will work solely on coding the project from the uml documents we will write up. The groups will work based on who has lab with whom (Carlos – Ben, Scott – Tucker) and the groups will be assigned different tasks at random.